Potential Field Navigation: Creating Potentials

ROB 102: Introduction to AI & Programming

Lecture 08

2021/10/20

Project 2: Potential Field Navigation

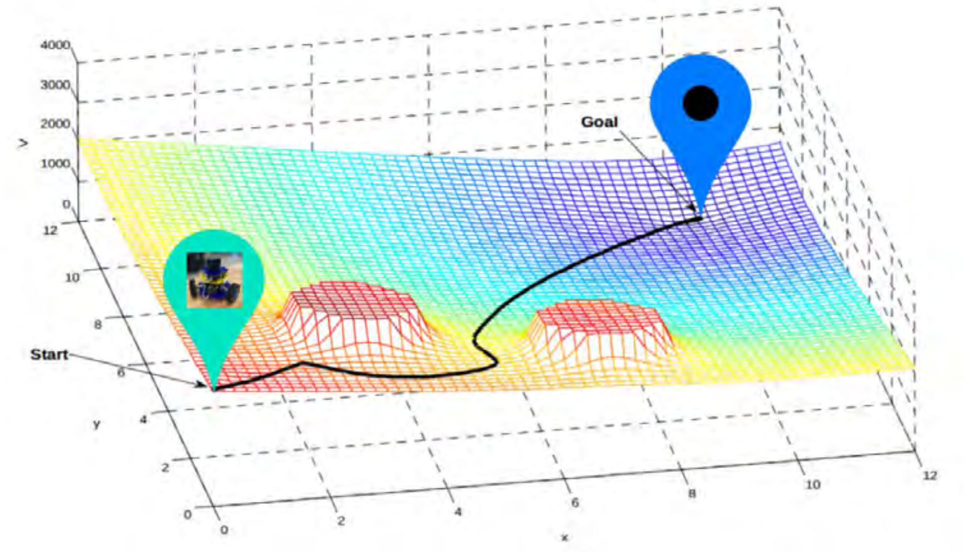
Lab 4

- ☑ Build a map of environment

Last lecture

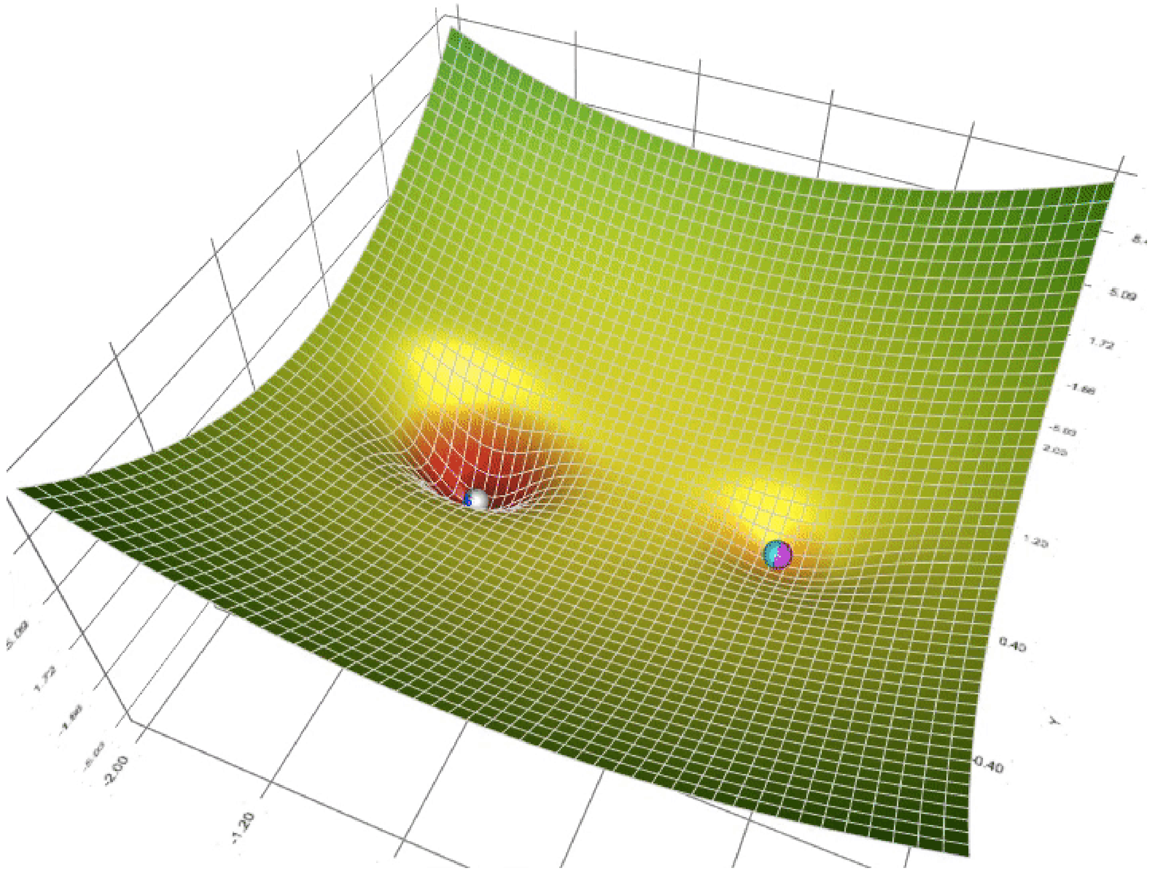
- ☑ Form attraction potential to goal
- ☑ Form repulsion potentials away from obstacles
- □ Add potentials together into potential field
- ☑ Local search over potential field to navigate

First potential field lecture



Last time…

(Link to GIF)



A **potential field** has *high* value in areas the robot should avoid and *low* value where the robot should go.

The robot navigates by moving to the area in its local region with the lowest potential.

Last time…

An **attractive potential** pulls the robot towards a goal.

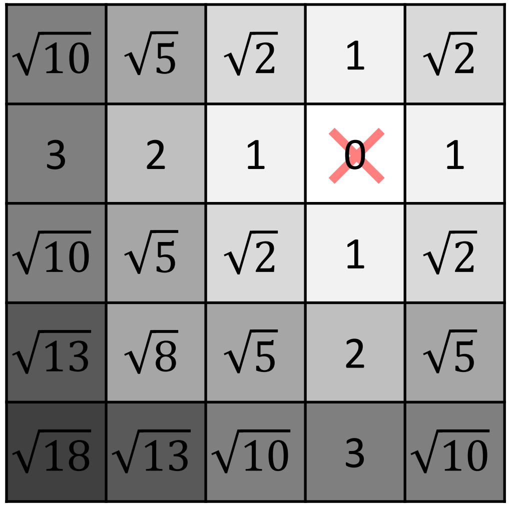
Attractive potential

$\sqrt{10}$	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$
3	2	1	0	1
$\sqrt{10}$	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$
$\sqrt{13}$	$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$
$\sqrt{18}$	$\sqrt{13}$	$\sqrt{10}$	3	$\sqrt{10}$

Last time…

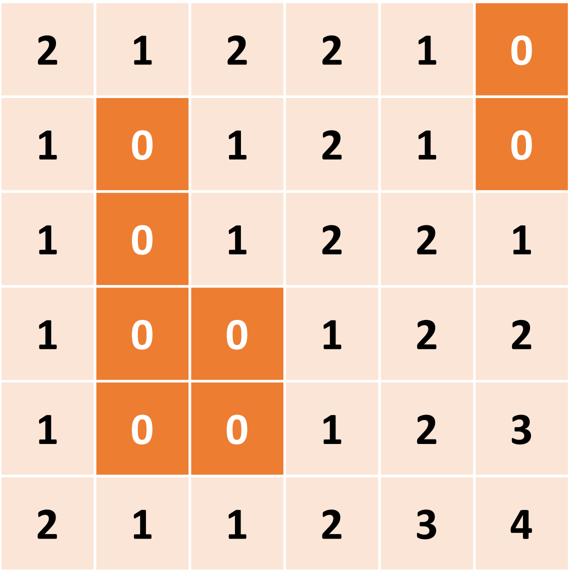
An **attractive potential** pulls the robot towards a goal.

A **distance transform** gives the distance to the nearest occupied cell for each cell.



$\sqrt{10}$	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$
3	2	1	$\xcancel{0}$	1
$\sqrt{10}$	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$
$\sqrt{13}$	$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$
$\sqrt{18}$	$\sqrt{13}$	$\sqrt{10}$	3	$\sqrt{10}$

Attractive potential



2	1	2	2	1	0
1	0	1	2	1	0
1	0	1	2	2	1
1	0	0	1	2	2
1	0	0	1	2	3
2	1	1	2	3	4

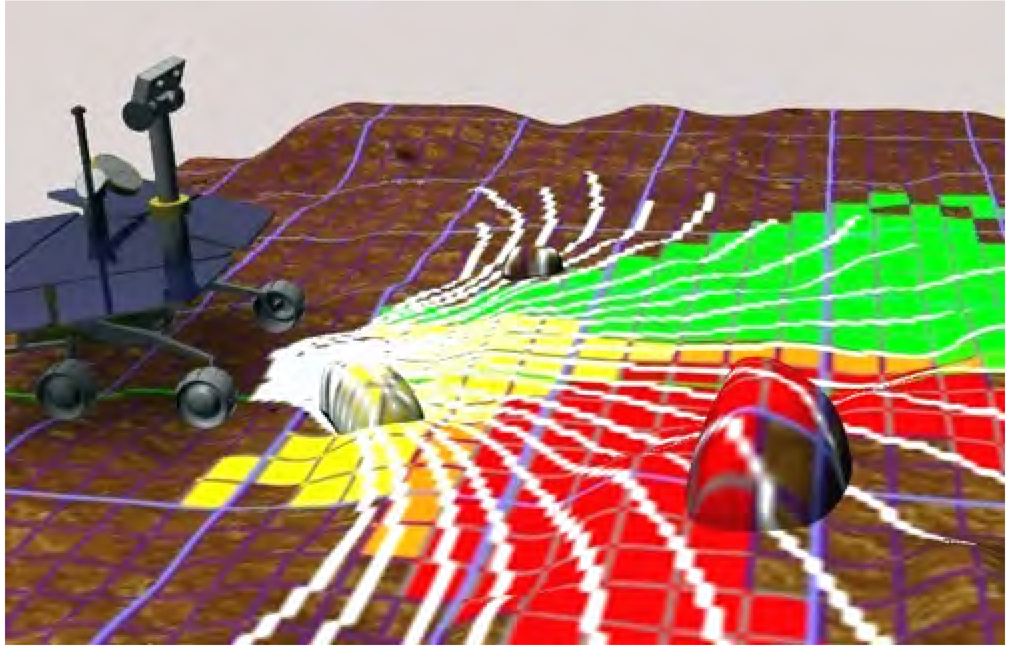
Manhattan distance transform

Potential Fields in the Wild

A rover can use potential field navigation to drive over rough terrain.

Potential field control is used for **local navigation**, to nearby goals.

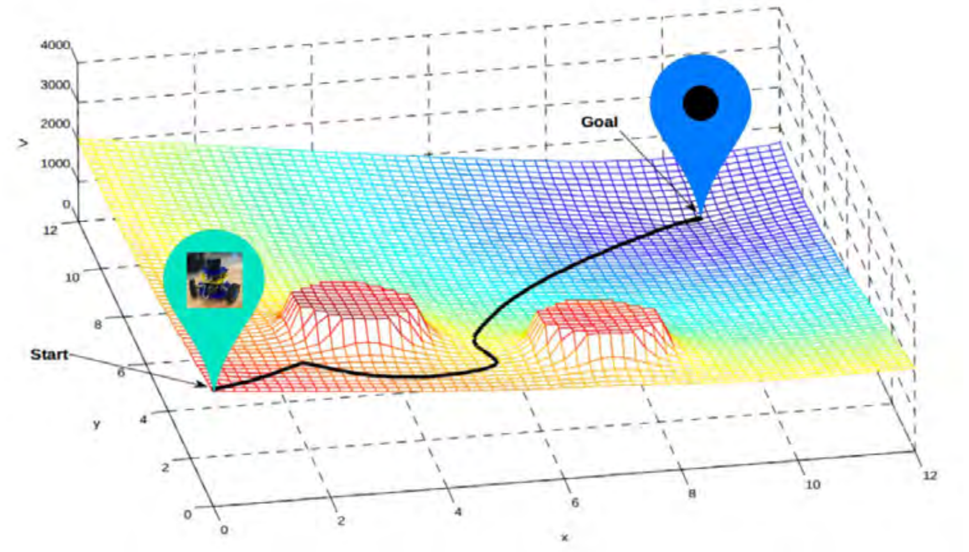
What about global navigation?



Global navigation: From any start to any goal, in any map

Project 2: Potential Field Navigation

- ☑ Build a map of environment
- ☑ Form attraction potential to goal
- ☑ Form repulsion potentials away from obstacles
- This lecture { ☐ Add potentials together into potential field
- ☑ Local search over potential field to navigate



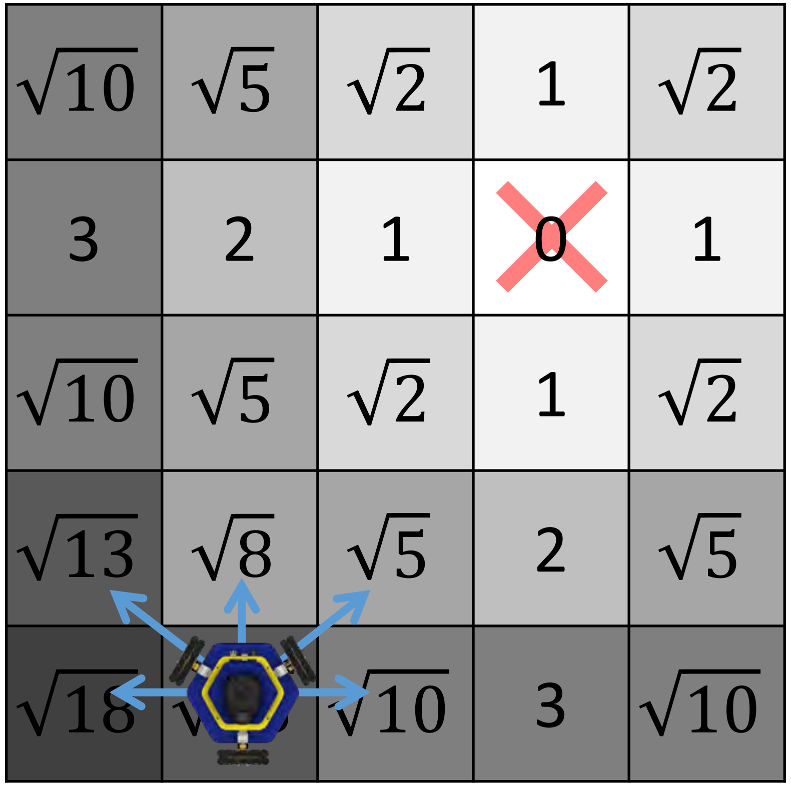
Attraction Potential & Local Search

An **attractive potential** *pulls* us towards a goal location.

We can think of the potential as the **potential energy** pulling the robot down a hill.

Attraction Potential & Local Search

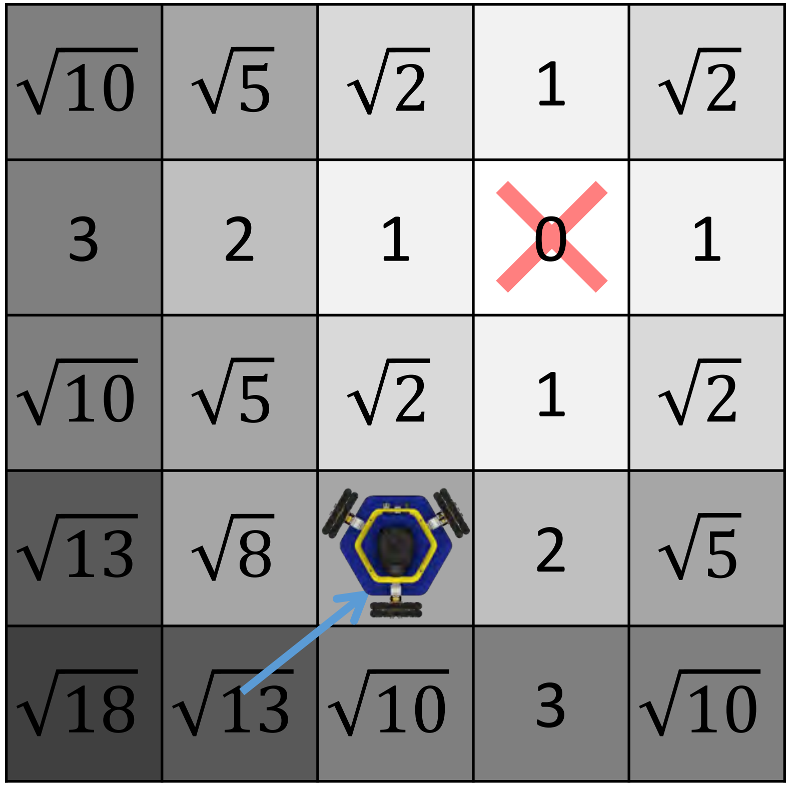
On a map, the **distance to the goal cell** is a useful attractive potential.



$\sqrt{10}$	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$
3	2	1	0	1
$\sqrt{10}$	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$
$\sqrt{13}$	$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$
$\sqrt{18}$	$\sqrt{13}$	$\sqrt{10}$	3	$\sqrt{10}$

Attraction Potential & Local Search

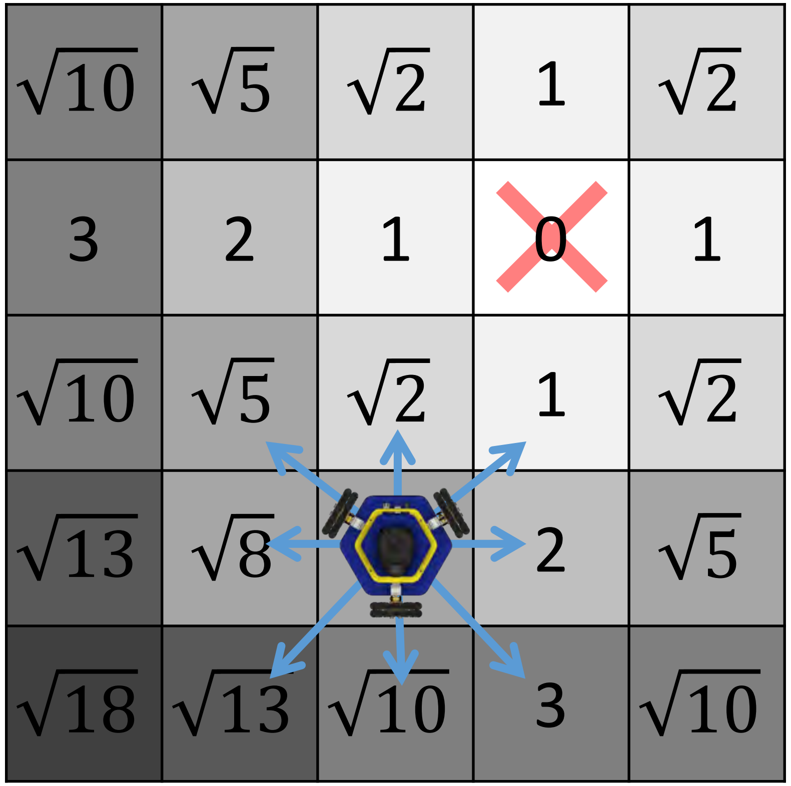
On a map, the **distance to the goal cell** is a useful attractive potential.



$\sqrt{10}$	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$
3	2	1	0	1
$\sqrt{10}$	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$
$\sqrt{13}$	$\sqrt{8}$		2	$\sqrt{5}$
$\sqrt{18}$	$\sqrt{13}$	$\sqrt{10}$	3	$\sqrt{10}$

Attraction Potential & Local Search

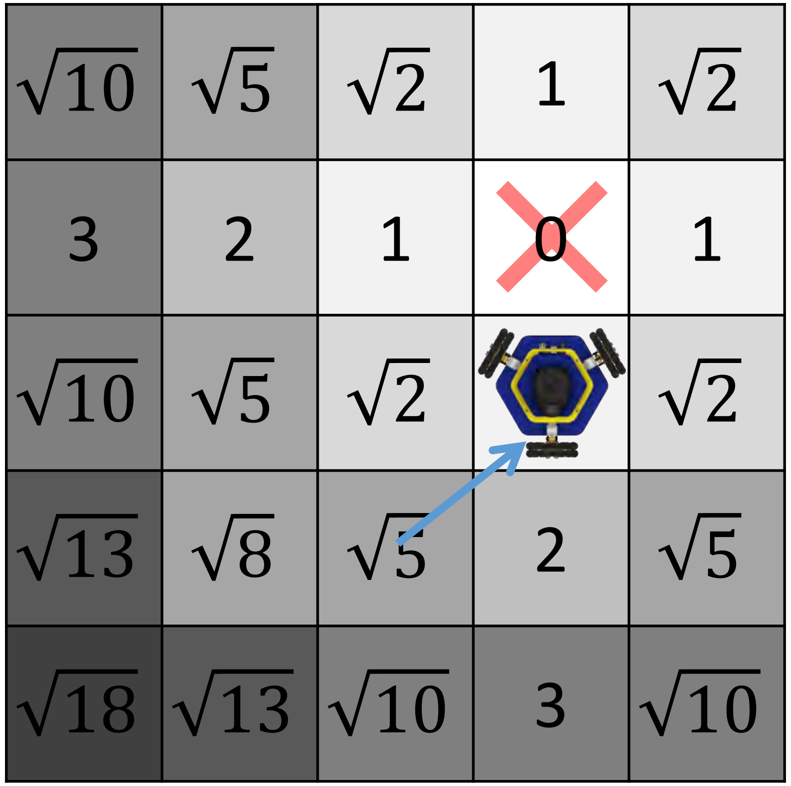
On a map, the **distance to the goal cell** is a useful attractive potential.



$\sqrt{10}$	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$
3	2	1	0	1
$\sqrt{10}$	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$
$\sqrt{13}$	$\sqrt{8}$		2	$\sqrt{5}$
$\sqrt{18}$	$\sqrt{13}$	$\sqrt{10}$	3	$\sqrt{10}$

Attraction Potential & Local Search

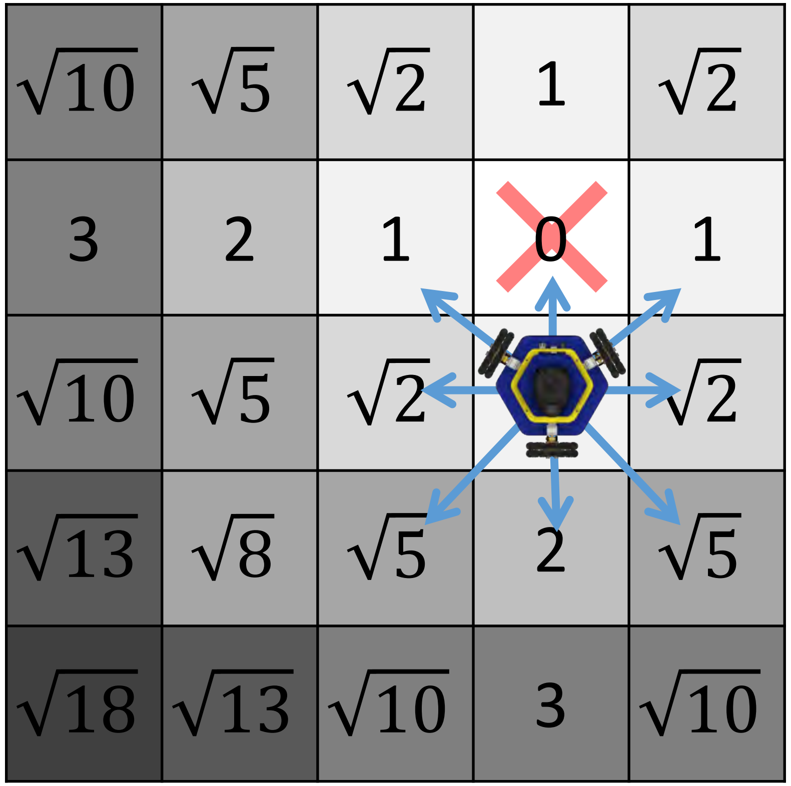
On a map, the **distance to the goal cell** is a useful attractive potential.



$\sqrt{10}$	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$
3	2	1	0	1
$\sqrt{10}$	$\sqrt{5}$	$\sqrt{2}$		$\sqrt{2}$
$\sqrt{13}$	$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$
$\sqrt{18}$	$\sqrt{13}$	$\sqrt{10}$	3	$\sqrt{10}$

Attraction Potential & Local Search

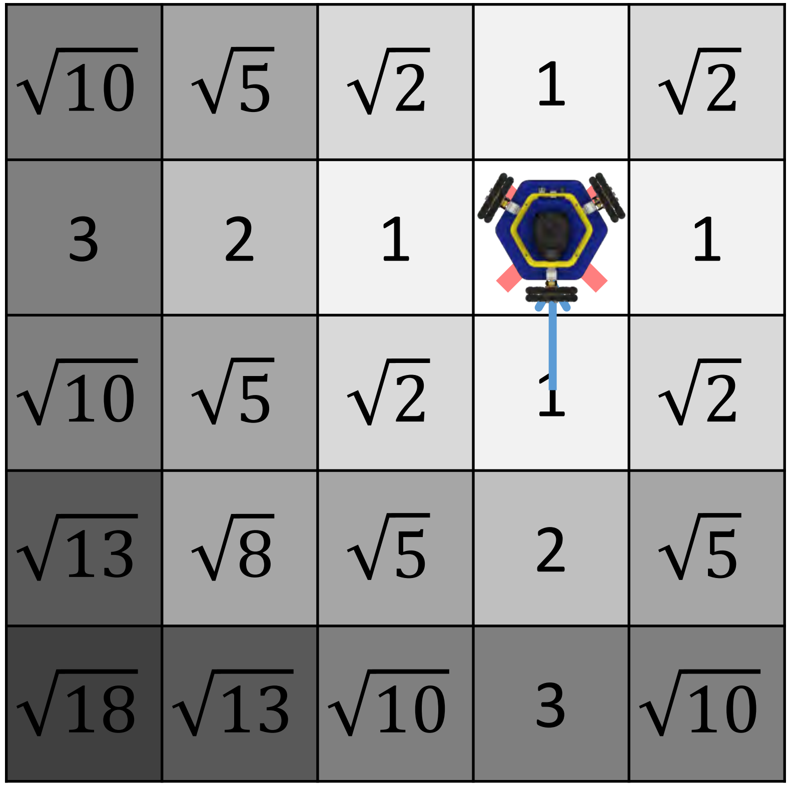
On a map, the **distance to the goal cell** is a useful attractive potential.



$\sqrt{10}$	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$
3	2	1	0	1
$\sqrt{10}$	$\sqrt{5}$	$\sqrt{2}$		$\sqrt{2}$
$\sqrt{13}$	$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$
$\sqrt{18}$	$\sqrt{13}$	$\sqrt{10}$	3	$\sqrt{10}$

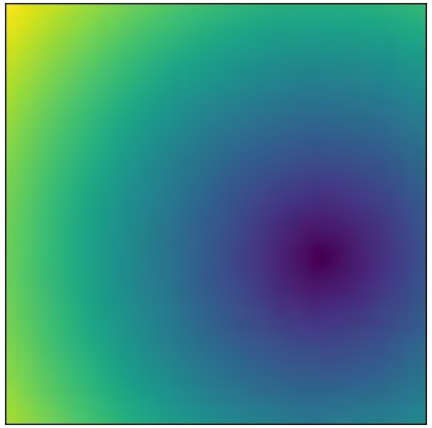
Attraction Potential & Local Search

On a map, the **distance to the goal cell** is a useful attractive potential.



$\sqrt{10}$	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$
3	2	1		1
$\sqrt{10}$	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$
$\sqrt{13}$	$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$
$\sqrt{18}$	$\sqrt{13}$	$\sqrt{10}$	3	$\sqrt{10}$

The Attraction Potential



Cone potential in 2D

Let's define two vectors:

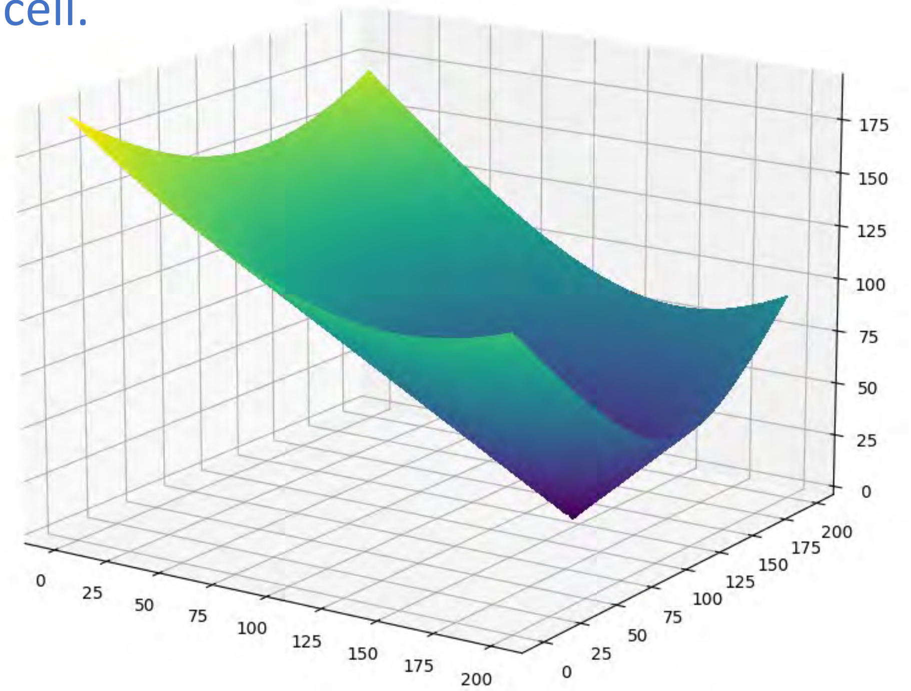
`goal_dists` The distance from each cell to the goal cell.

`attract_field` The attraction potential at each cell.

Let's define the potential at cell `i` as:

```
attract_field[i] = goal_dists[i]
```

This defines a cone potential.



The Attraction Potential

Let’s define two vectors:

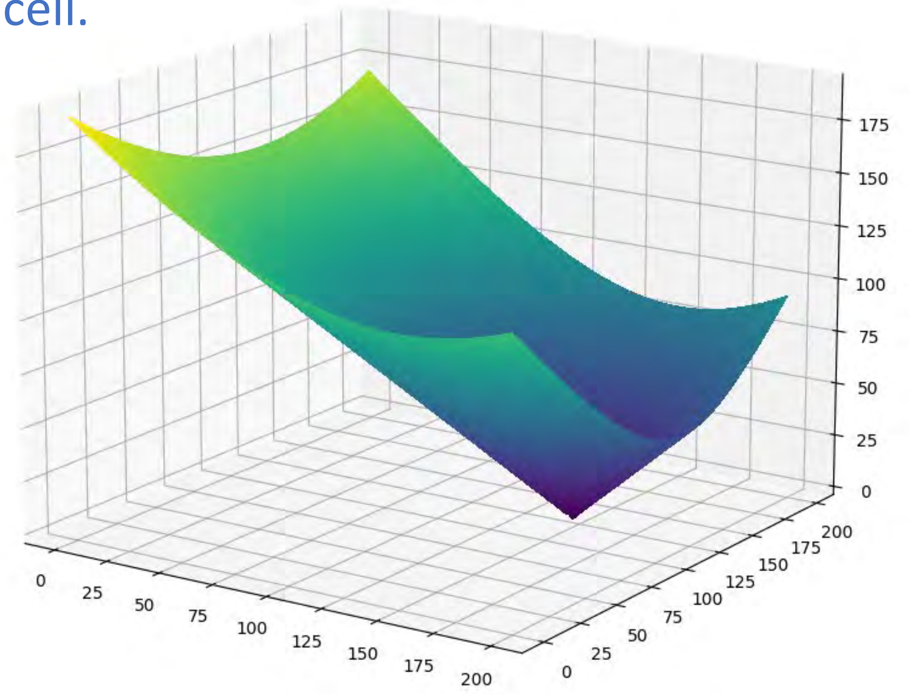
`goal_dists` The distance from each cell to the goal cell.

`attract_field` The attraction potential at each cell.

Let’s define the potential at cell `i` as:

```
attract_field[i] = goal_dists[i]
```

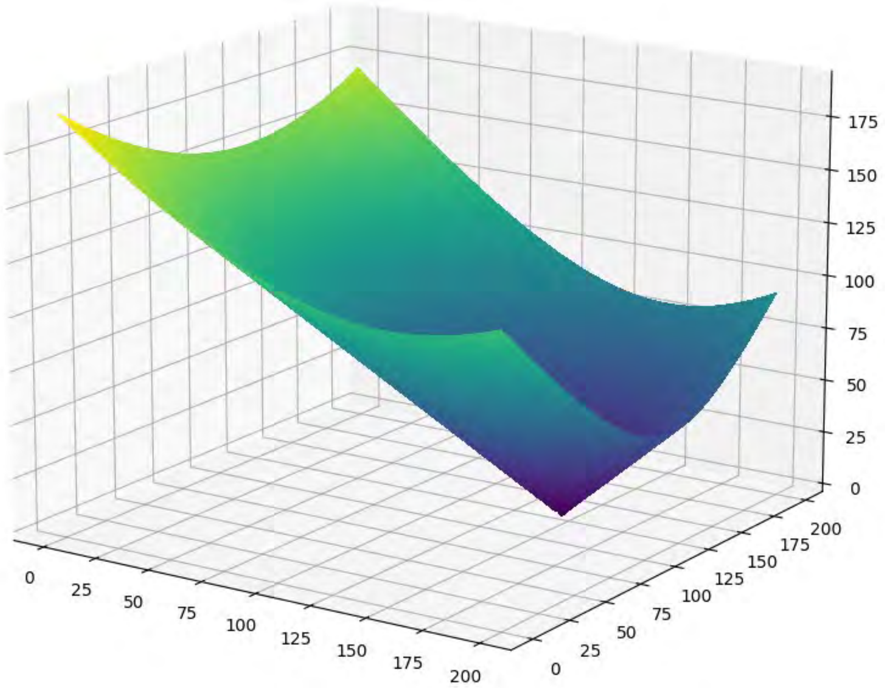
This defines a cone potential.



The Attraction Potential

The cone potential:

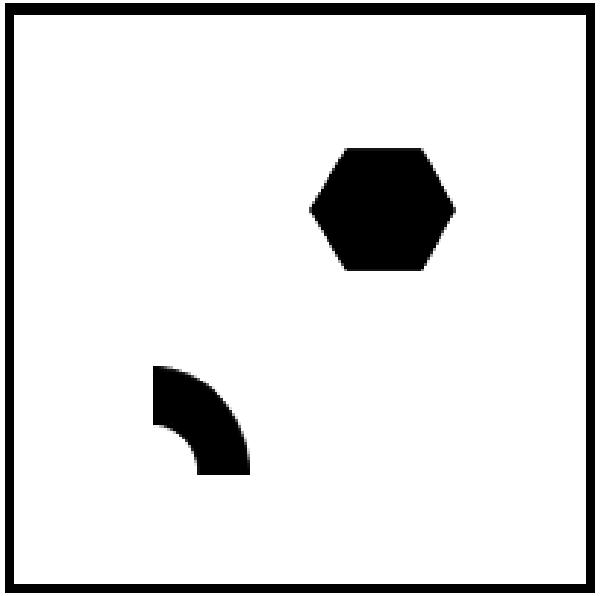
```
attract_field[i] = goal_dists[i]
```



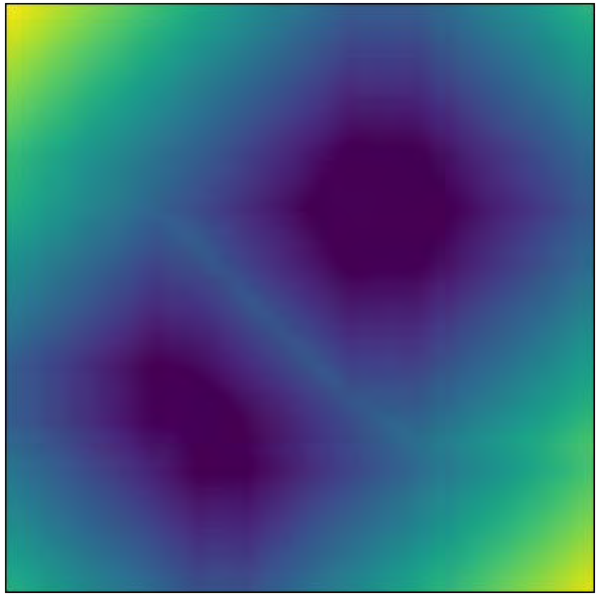
Problem: Doesn’t avoid obstacles.

The Distance Transform

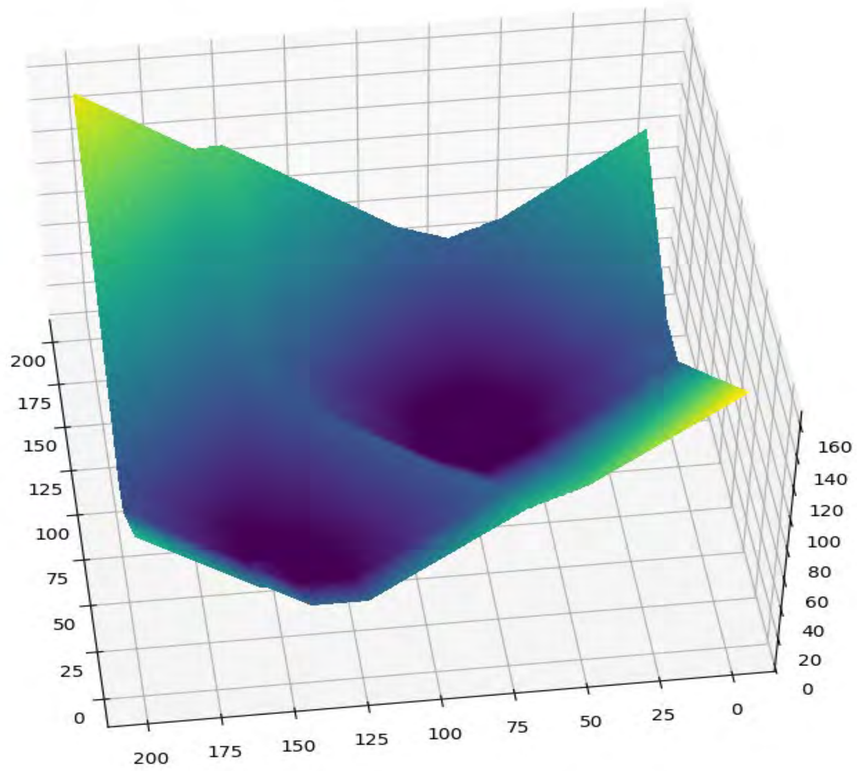
The **distance transform** gives the distance to the nearest obstacle at each cell.



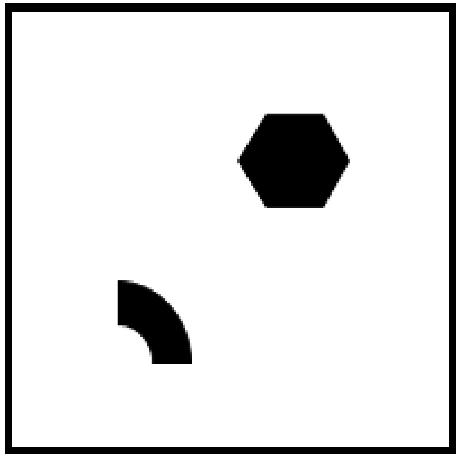
Binary Image



2D Distance Transform

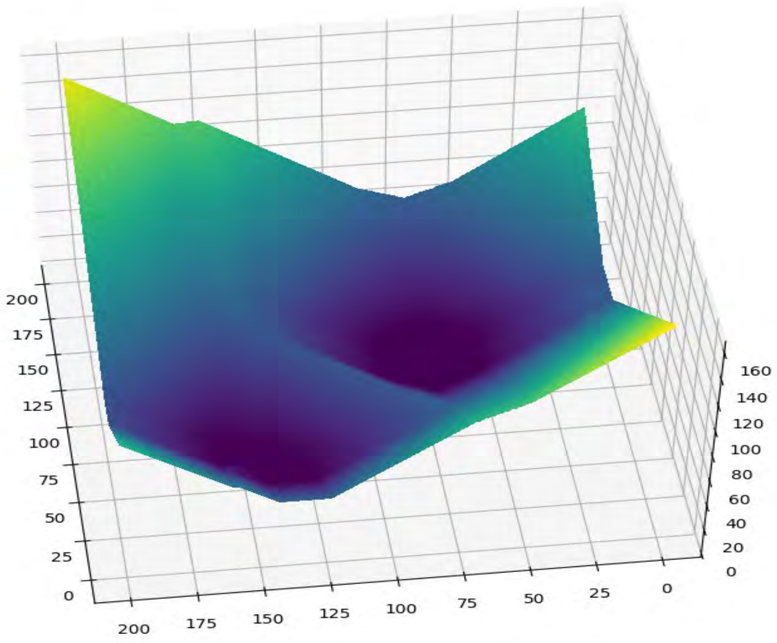


The Distance Transform



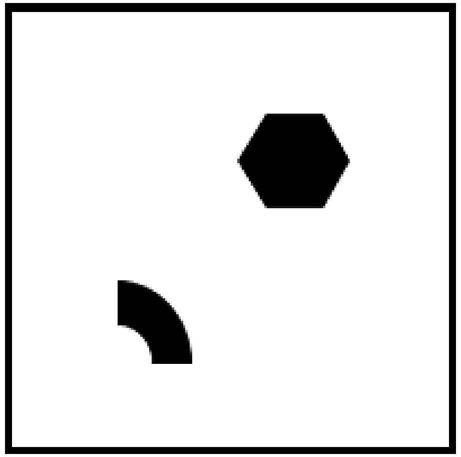
The distance transform is LOW where the robot should avoid and HIGH where the robot should go.

This is the opposite of the repulsive field.



Distance Transform

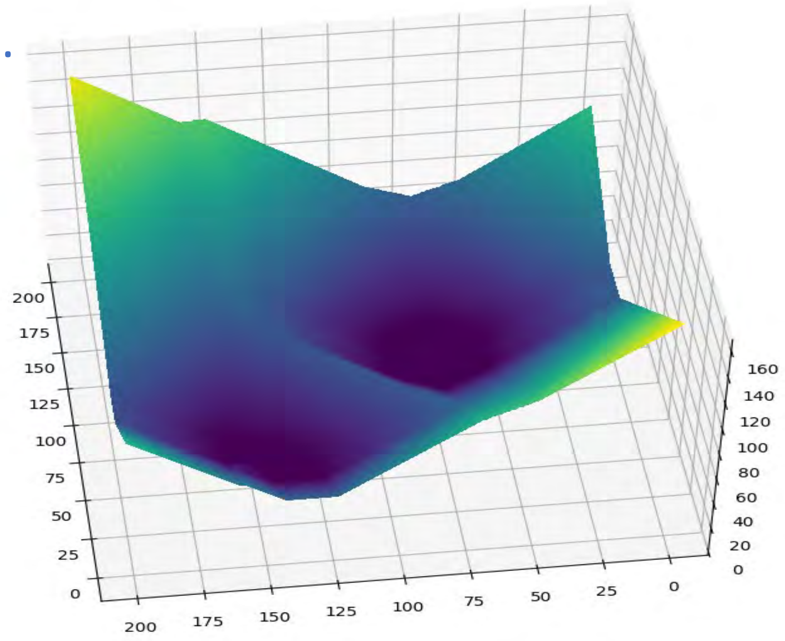
The Repulsion Potential



Let's define two vectors:

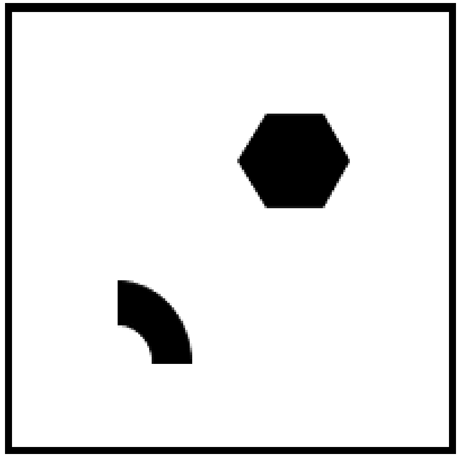
`obstacle_dists` The distance from each cell to the nearest obstacle.

`repul_field` The repulsion potential at each cell.



Distance Transform

The Repulsion Potential



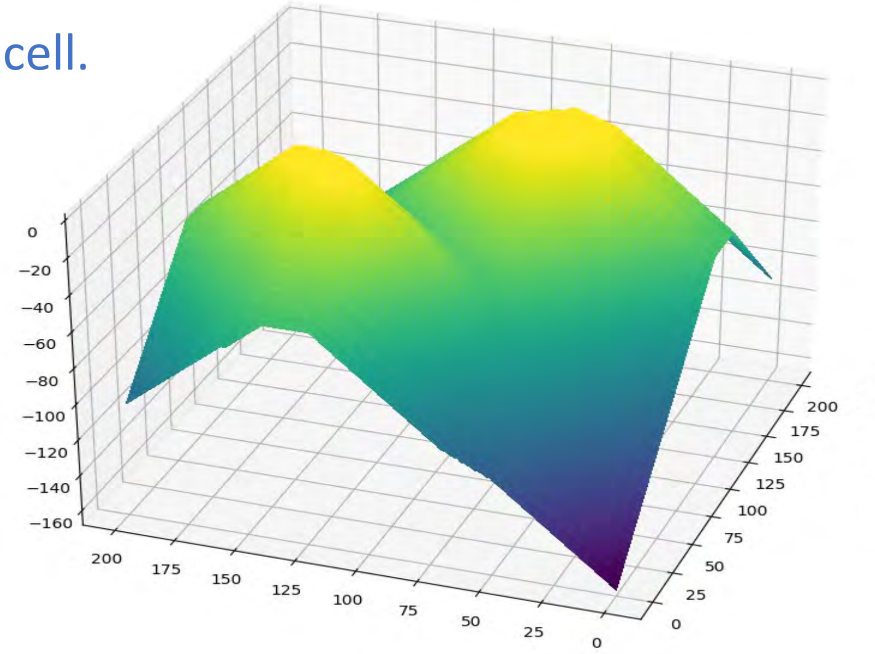
Let's define two vectors:

`obstacle_dists` The distance from each cell to the nearest obstacle.

`repul_field` The repulsion potential at each cell.

Idea: The repulsion field is the negative of the distance transform.

```
repul_field[i] = -obstacle_dists[i]
```



Repulsion Potential

Combining Potential Fields

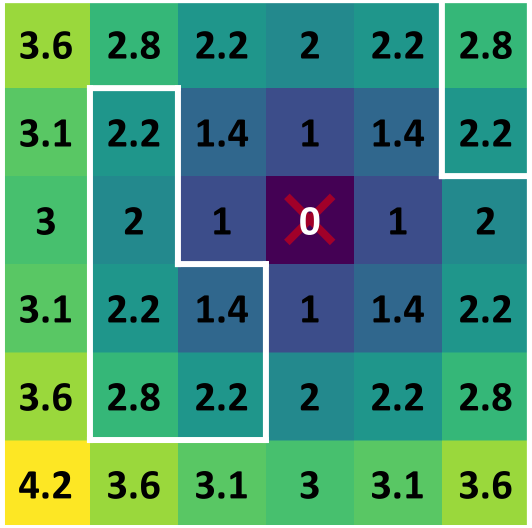
Idea: The potential field is the attractive field plus the repulsive field.

```
potential[i] = attract_field[i] + repul_field[i]
```

Combining Potential Fields

Idea: The potential field is the attractive field plus the repulsive field.

```
potential[i] = attract_field[i] + repul_field[i]
```



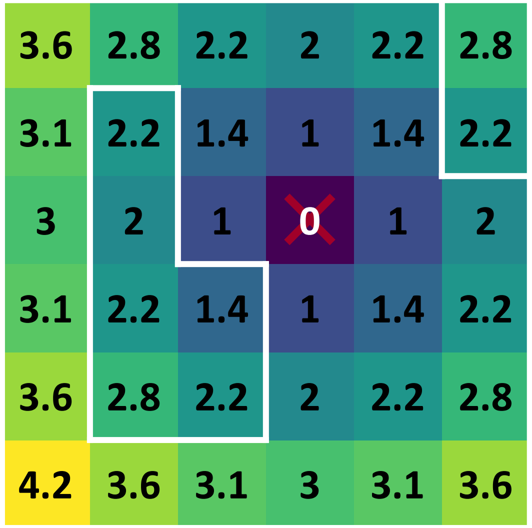
3.6	2.8	2.2	2	2.2	2.8
3.1	2.2	1.4	1	1.4	2.2
3	2	1	0	1	2
3.1	2.2	1.4	1	1.4	2.2
3.6	2.8	2.2	2	2.2	2.8
4.2	3.6	3.1	3	3.1	3.6

```
attract_field = goal_dists
```

Combining Potential Fields

Idea: The potential field is the attractive field plus the repulsive field.

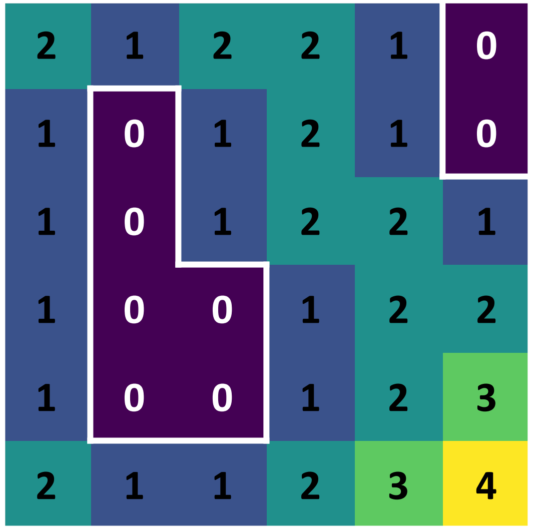
```
potential[i] = attract_field[i] + repul_field[i]
```



3.6	2.8	2.2	2	2.2	2.8
3.1	2.2	1.4	1	1.4	2.2
3	2	1	0	1	2
3.1	2.2	1.4	1	1.4	2.2
3.6	2.8	2.2	2	2.2	2.8
4.2	3.6	3.1	3	3.1	3.6

attract_field

+ -1*



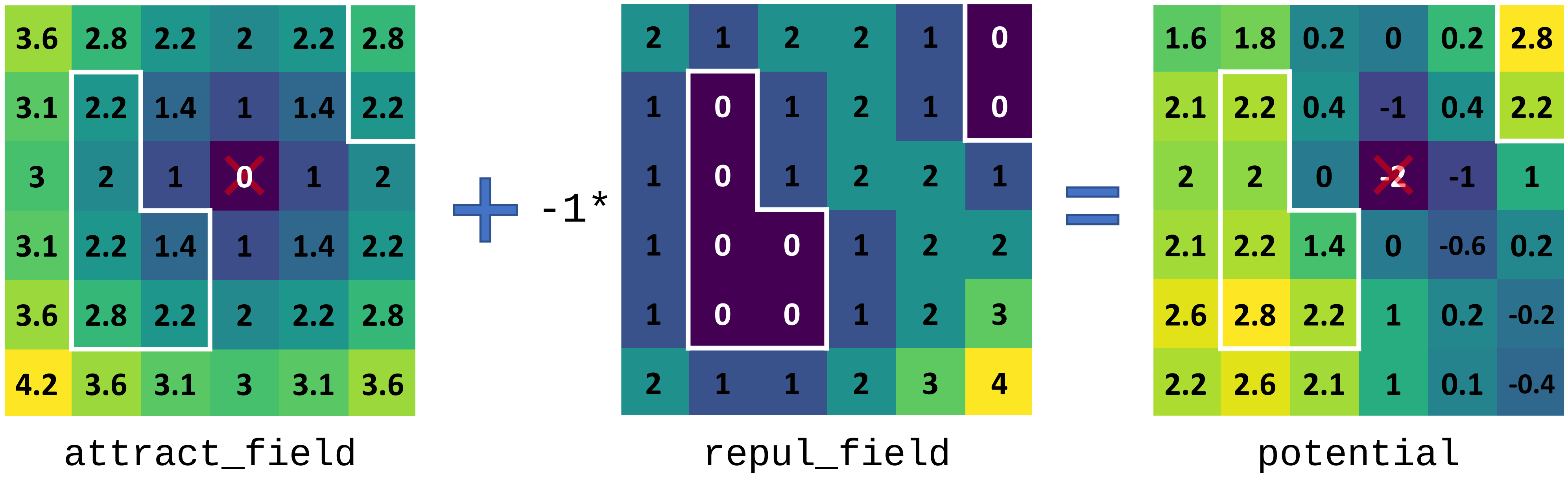
2	1	2	2	1	0
1	0	1	2	1	0
1	0	1	2	2	1
1	0	0	1	2	2
1	0	0	1	2	3
2	1	1	2	3	4

repul_field = -obstacle_dists

Combining Potential Fields

Idea: The potential field is the attractive field plus the repulsive field.

```
potential[i] = attract_field[i] + repul_field[i]
```



3.6	2.8	2.2	2	2.2	2.8
3.1	2.2	1.4	1	1.4	2.2
3	2	1	0	1	2
3.1	2.2	1.4	1	1.4	2.2
3.6	2.8	2.2	2	2.2	2.8
4.2	3.6	3.1	3	3.1	3.6

`attract_field`

+ -1*

2	1	2	2	1	0
1	0	1	2	1	0
1	0	1	2	2	1
1	0	0	1	2	2
1	0	0	1	2	3
2	1	1	2	3	4

`repul_field`

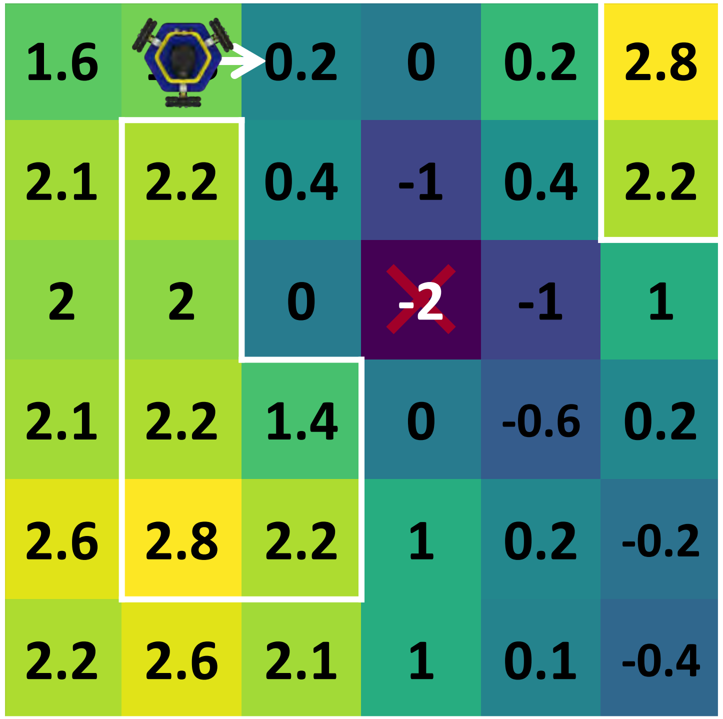
=

1.6	1.8	0.2	0	0.2	2.8
2.1	2.2	0.4	-1	0.4	2.2
2	2	0	-2	-1	1
2.1	2.2	1.4	0	-0.6	0.2
2.6	2.8	2.2	1	0.2	-0.2
2.2	2.6	2.1	1	0.1	-0.4

`potential`

Combining Potential Fields

Let’s take a closer look at our potential field.

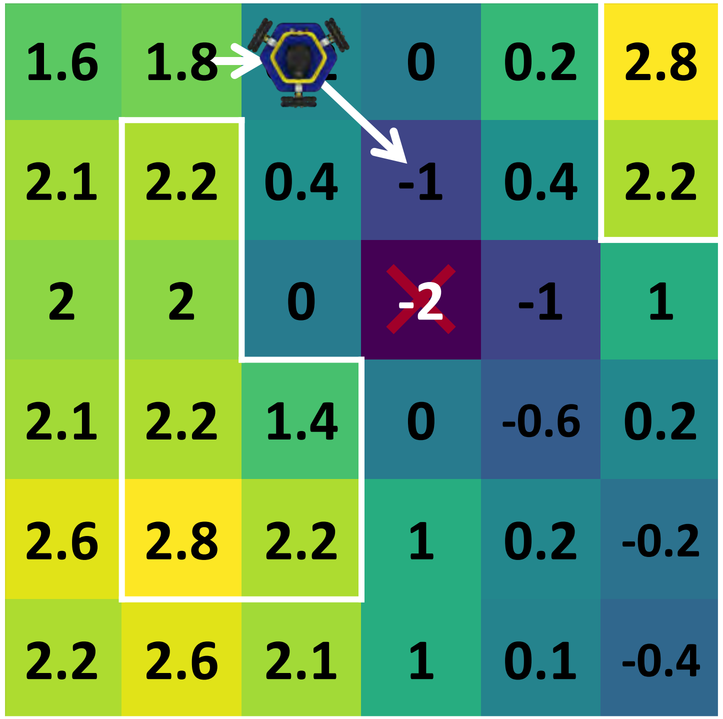


1.6	[illegible]	0.2	0	0.2	2.8
2.1	2.2	0.4	-1	0.4	2.2
2	2	0	-2	-1	1
2.1	2.2	1.4	0	-0.6	0.2
2.6	2.8	2.2	1	0.2	-0.2
2.2	2.6	2.1	1	0.1	-0.4

`potential`

Combining Potential Fields

Let’s take a closer look at our potential field.

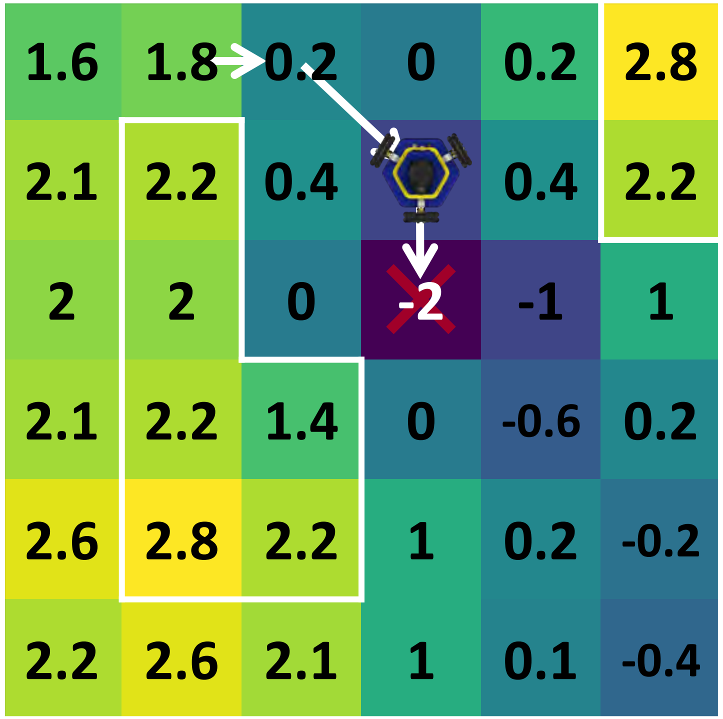


1.6	1.8	[illegible]	0	0.2	2.8
2.1	2.2	0.4	-1	0.4	2.2
2	2	0	-2	-1	1
2.1	2.2	1.4	0	-0.6	0.2
2.6	2.8	2.2	1	0.2	-0.2
2.2	2.6	2.1	1	0.1	-0.4

potential

Combining Potential Fields

Let’s take a closer look at our potential field.

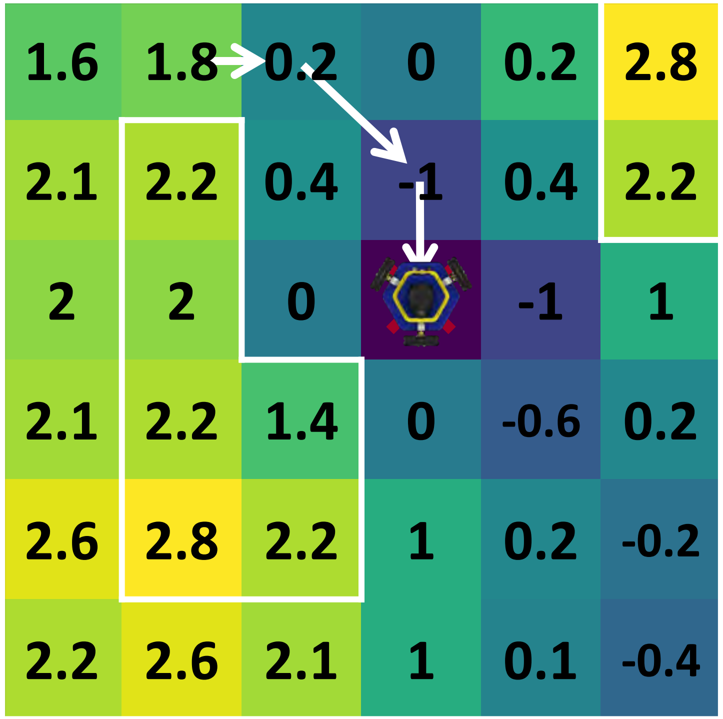


1.6	1.8	0.2	0	0.2	2.8
2.1	2.2	0.4		0.4	2.2
2	2	0	-2	-1	1
2.1	2.2	1.4	0	-0.6	0.2
2.6	2.8	2.2	1	0.2	-0.2
2.2	2.6	2.1	1	0.1	-0.4

potential

Combining Potential Fields

Let’s take a closer look at our potential field.



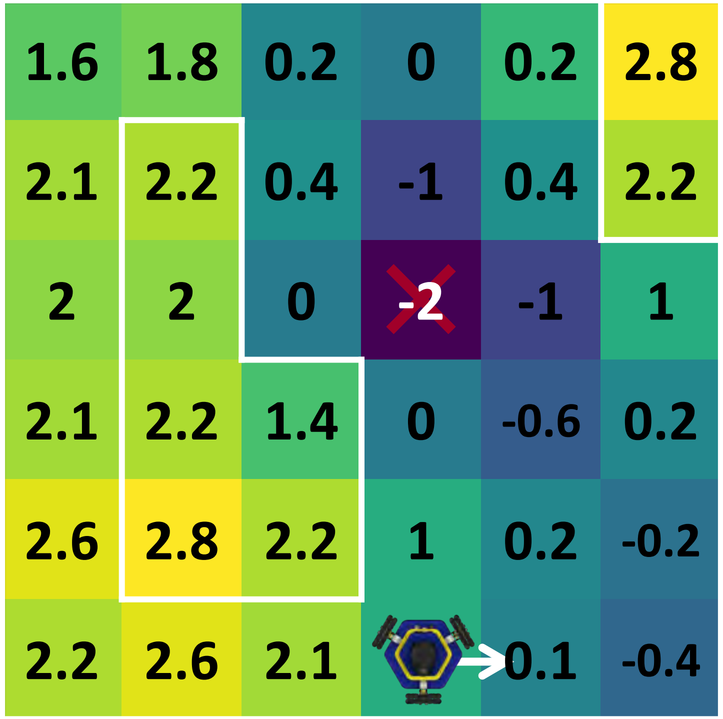
1.6	1.8	0.2	0	0.2	2.8
2.1	2.2	0.4	-1	0.4	2.2
2	2	0		-1	1
2.1	2.2	1.4	0	-0.6	0.2
2.6	2.8	2.2	1	0.2	-0.2
2.2	2.6	2.1	1	0.1	-0.4

`potential`

Success!

Combining Potential Fields

Let’s take a closer look at our potential field.

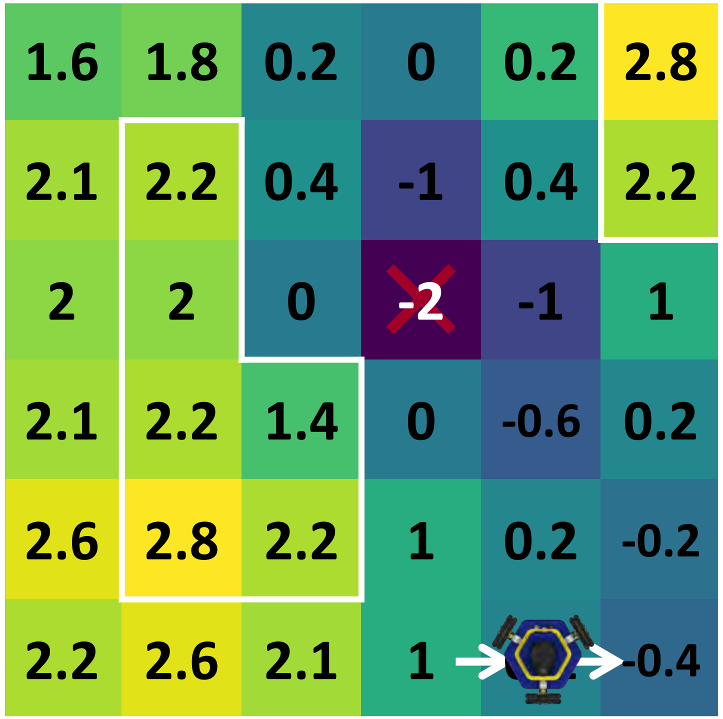


1.6	1.8	0.2	0	0.2	2.8
2.1	2.2	0.4	-1	0.4	2.2
2	2	0	-2	-1	1
2.1	2.2	1.4	0	-0.6	0.2
2.6	2.8	2.2	1	0.2	-0.2
2.2	2.6	2.1		0.1	-0.4

`potential`

Combining Potential Fields

Let’s take a closer look at our potential field.

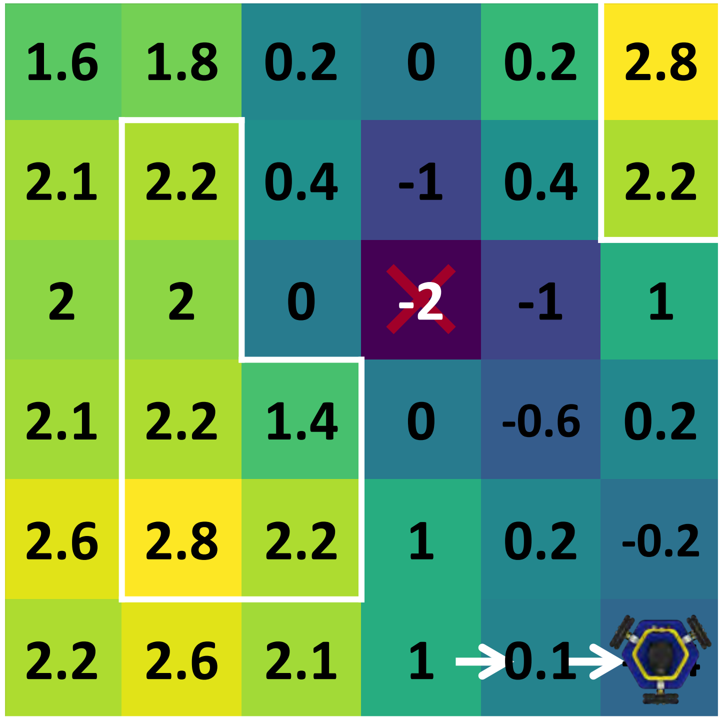


1.6	1.8	0.2	0	0.2	2.8
2.1	2.2	0.4	-1	0.4	2.2
2	2	0	-2	-1	1
2.1	2.2	1.4	0	-0.6	0.2
2.6	2.8	2.2	1	0.2	-0.2
2.2	2.6	2.1	1	0.2	-0.4

potential

Combining Potential Fields

Let’s take a closer look at our potential field.



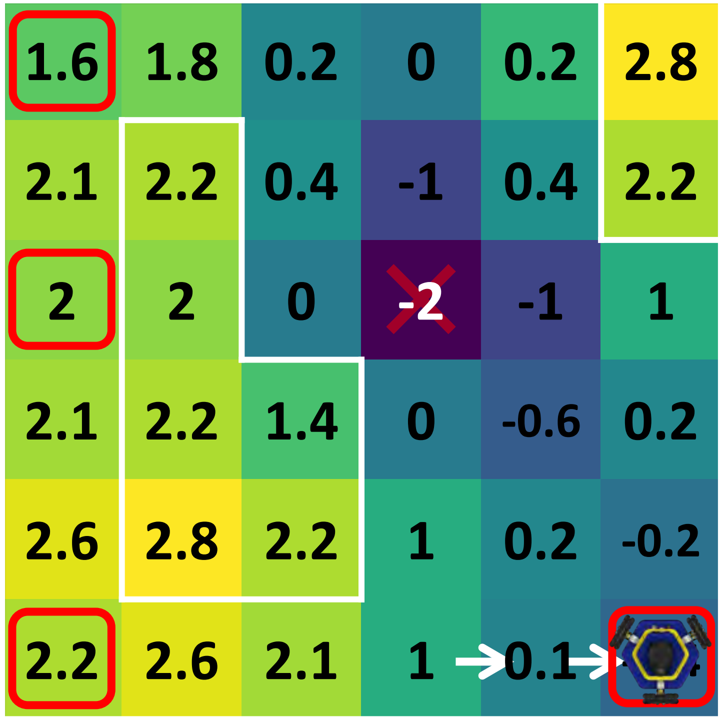
1.6	1.8	0.2	0	0.2	2.8
2.1	2.2	0.4	-1	0.4	2.2
2	2	0	-2	-1	1
2.1	2.2	1.4	0	-0.6	0.2
2.6	2.8	2.2	1	0.2	-0.2
2.2	2.6	2.1	1	0.1	[illegible]

Local Minimum :(

potential

Combining Potential Fields

Let’s take a closer look at our potential field.



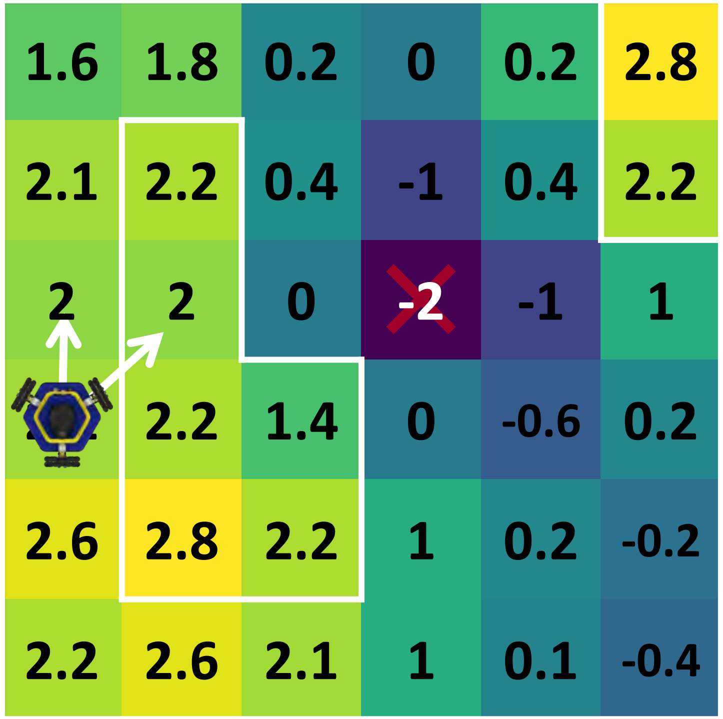
1.6	1.8	0.2	0	0.2	2.8
2.1	2.2	0.4	-1	0.4	2.2
2	2	0	-2	-1	1
2.1	2.2	1.4	0	-0.6	0.2
2.6	2.8	2.2	1	0.2	-0.2
2.2	2.6	2.1	1 →	0.1 →	[illegible]

potential

Lots of local minima!

Combining Potential Fields

Let’s take a closer look at our potential field.

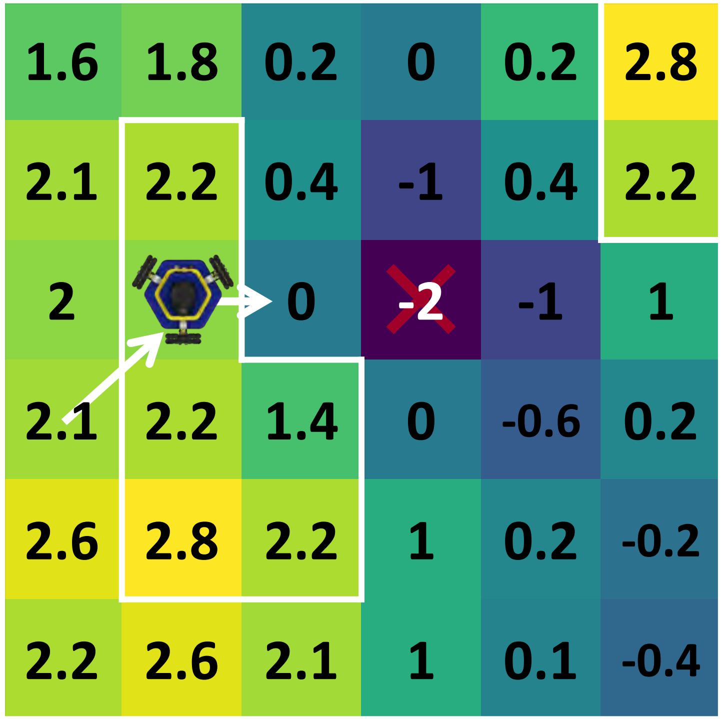


1.6	1.8	0.2	0	0.2	2.8
2.1	2.2	0.4	-1	0.4	2.2
2	2	0	-2	-1	1
[illegible]	2.2	1.4	0	-0.6	0.2
2.6	2.8	2.2	1	0.2	-0.2
2.2	2.6	2.1	1	0.1	-0.4

`potential`

Combining Potential Fields

Let’s take a closer look at our potential field.



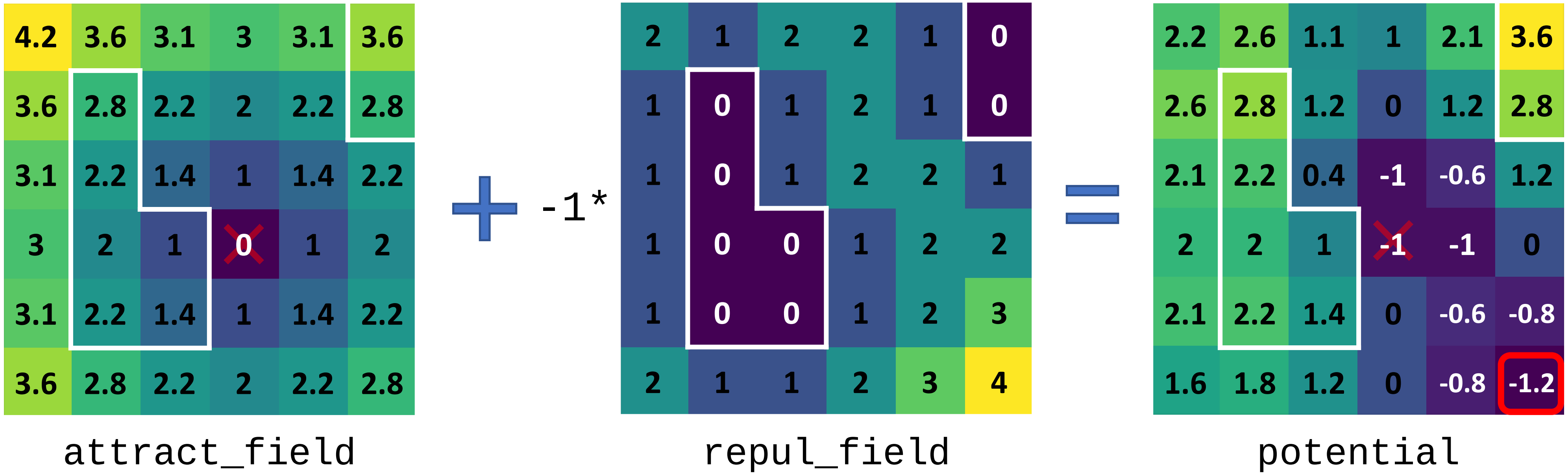
1.6	1.8	0.2	0	0.2	2.8
2.1	2.2	0.4	-1	0.4	2.2
2		0	-2	-1	1
2.1	2.2	1.4	0	-0.6	0.2
2.6	2.8	2.2	1	0.2	-0.2
2.2	2.6	2.1	1	0.1	-0.4

Collision!!

potential

Combining Potential Fields

Depending on the goal, the lowest point of the potential field might not be at the goal location!



4.2	3.6	3.1	3	3.1	3.6
3.6	2.8	2.2	2	2.2	2.8
3.1	2.2	1.4	1	1.4	2.2
3	2	1	0	1	2
3.1	2.2	1.4	1	1.4	2.2
3.6	2.8	2.2	2	2.2	2.8

attract_field

+ -1*

2	1	2	2	1	0
1	0	1	2	1	0
1	0	1	2	2	1
1	0	0	1	2	2
1	0	0	1	2	3
2	1	1	2	3	4

repul_field

=

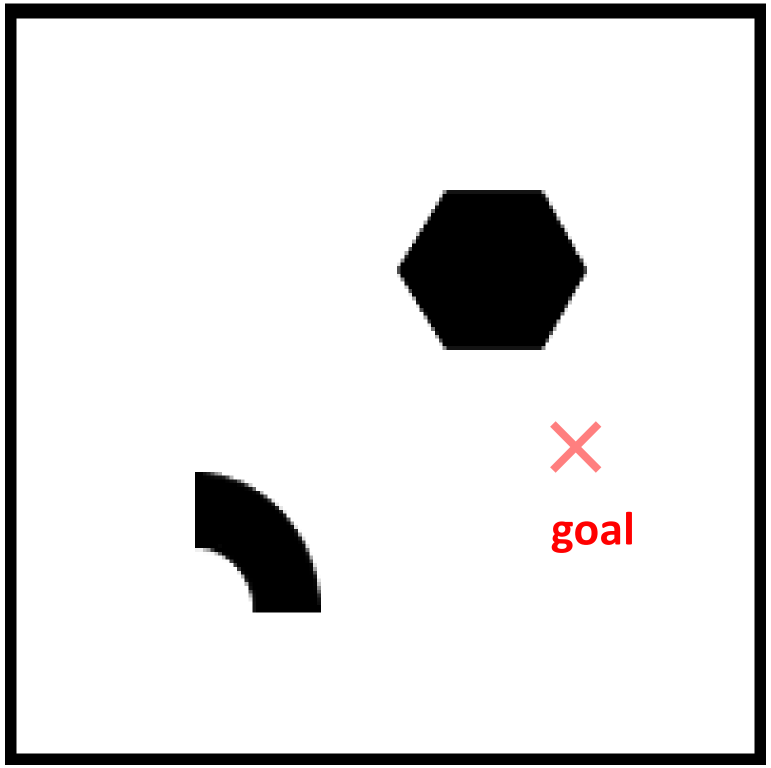
2.2	2.6	1.1	1	2.1	3.6
2.6	2.8	1.2	0	1.2	2.8
2.1	2.2	0.4	-1	-0.6	1.2
2	2	1	-1	-1	0
2.1	2.2	1.4	0	-0.6	-0.8
1.6	1.8	1.2	0	-0.8	-1.2

potential

Computing the Potential Field

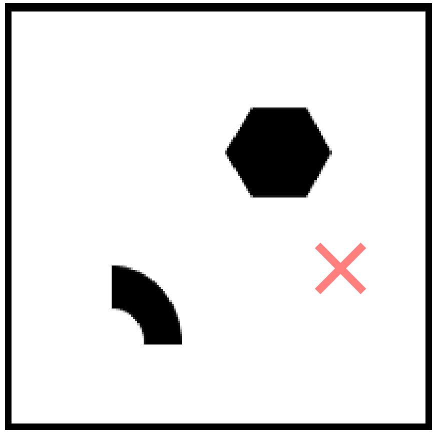
A larger example:

two_obstacles.map

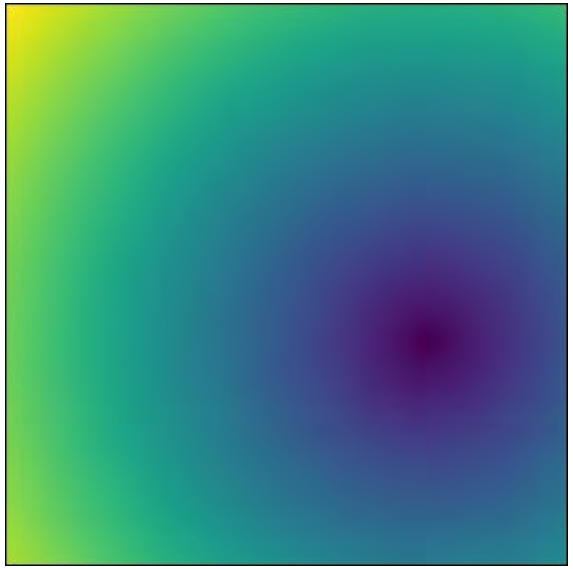


Computing the Potential Field

two_obstacles.map

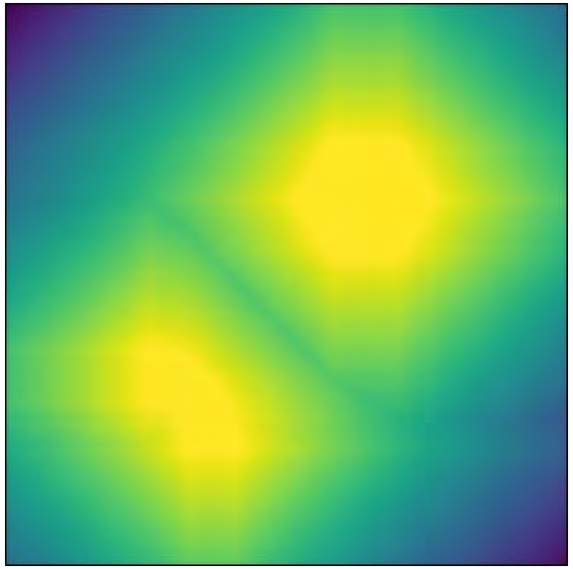


A larger example:



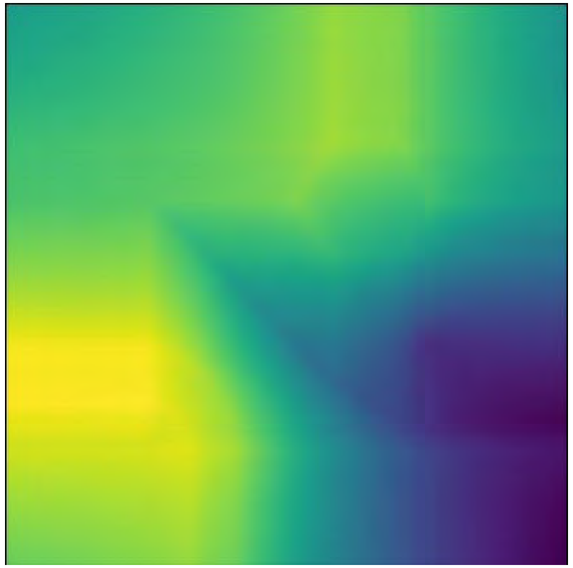
attract_field

+



repul_field

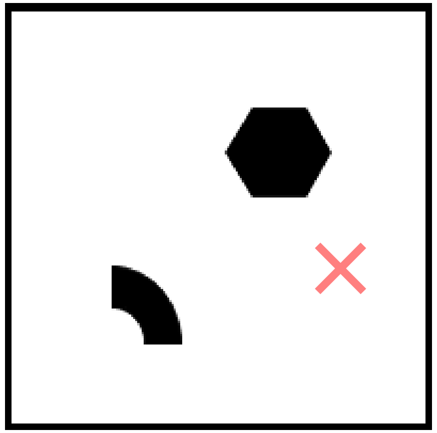
=



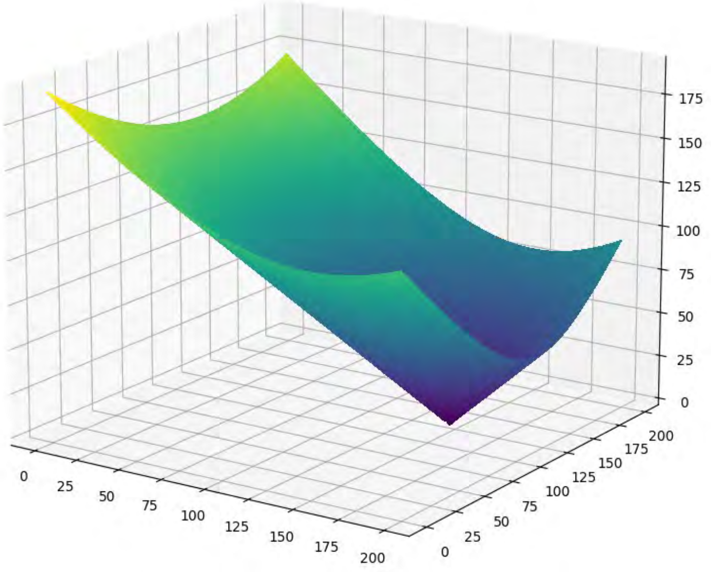
potential

Computing the Potential Field

two_obstacles.map

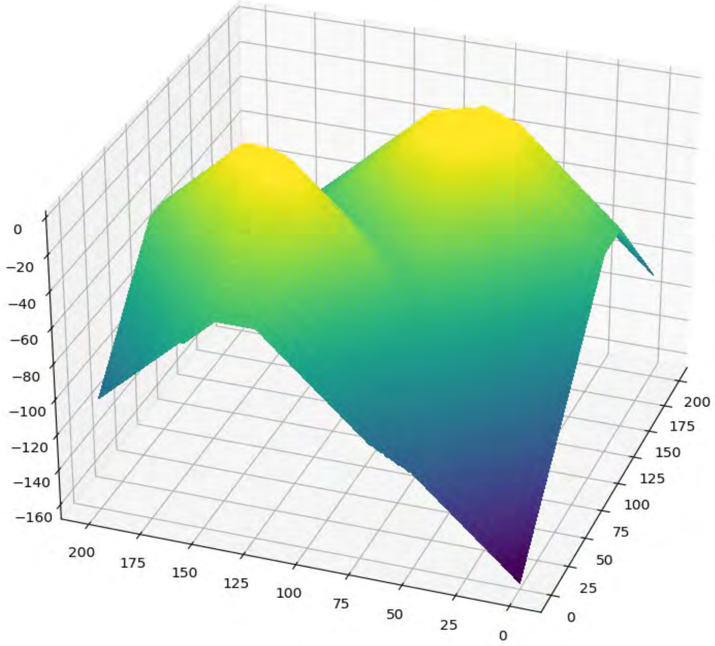


A larger example:



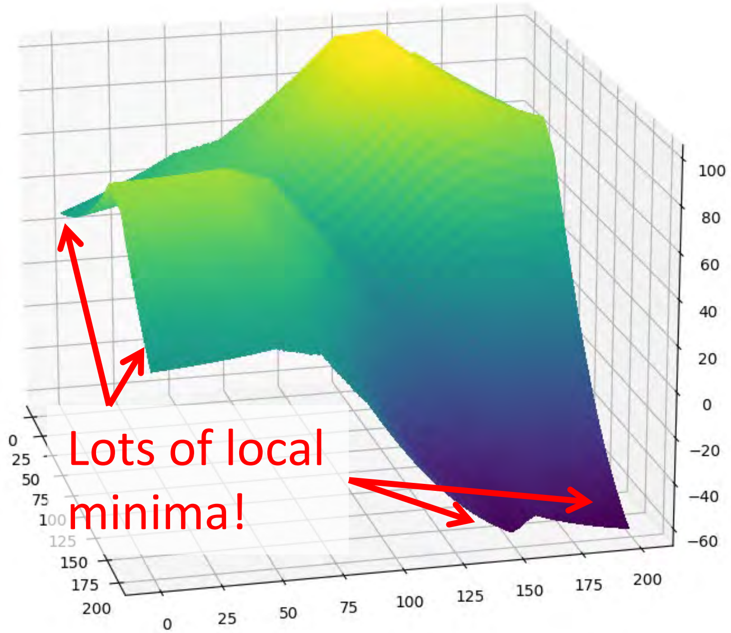
attract_field

+



repul_field

=

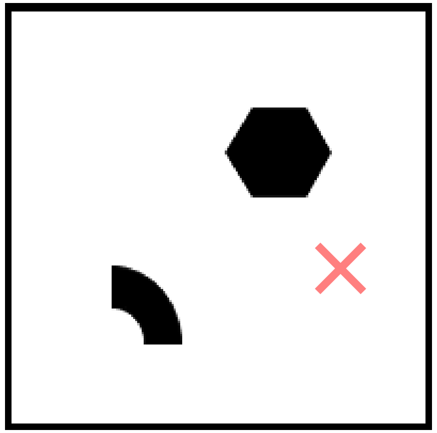


Lots of local minima!

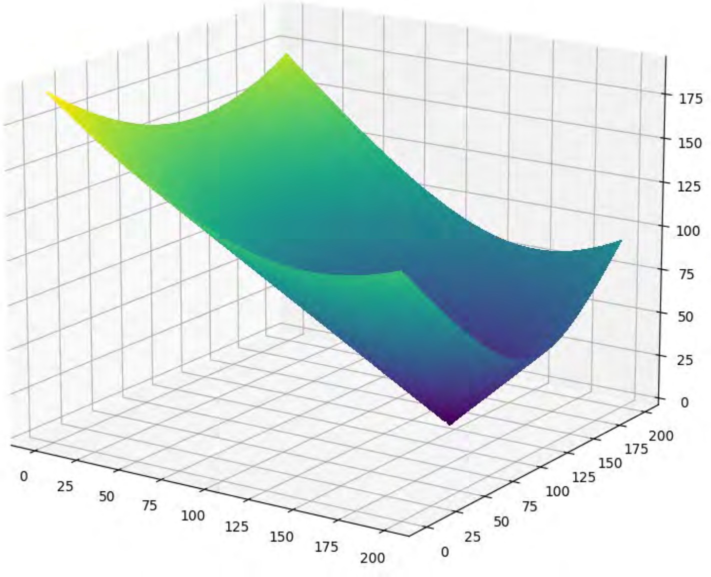
potential

Computing the Potential Field

two_obstacles.map

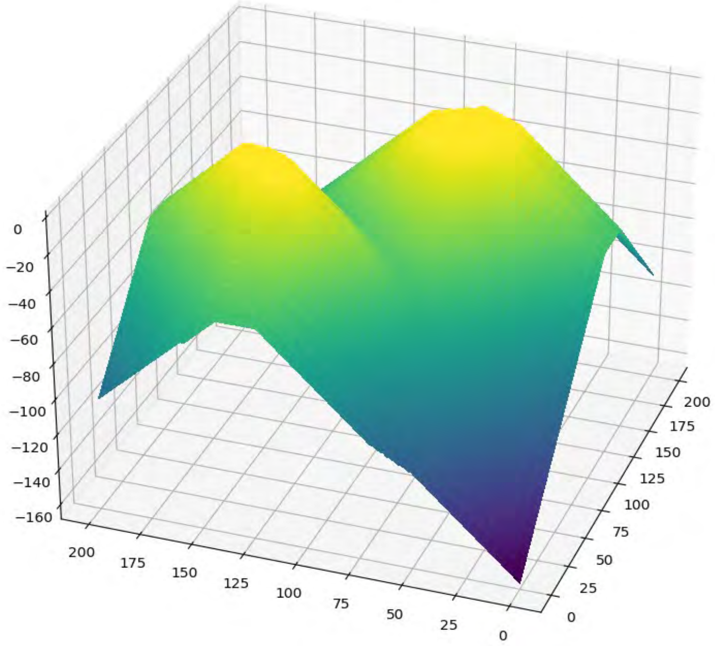


Can we do better?



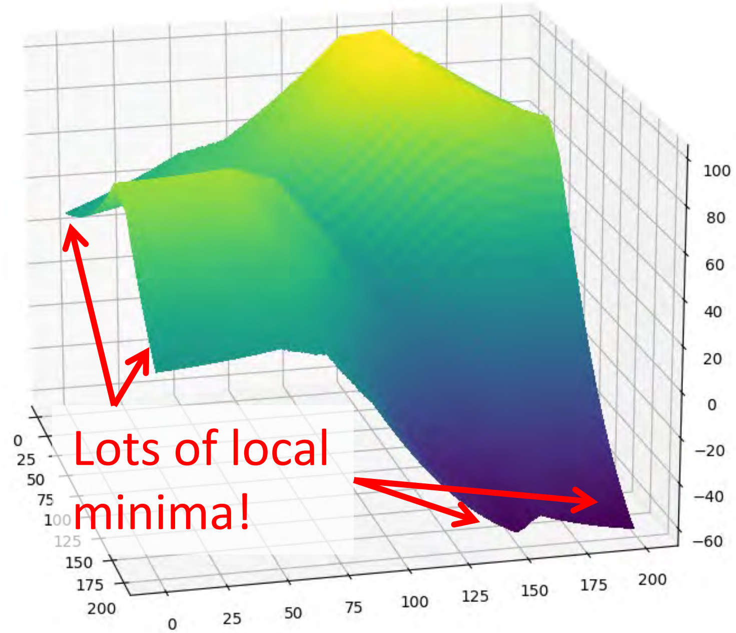
attract_field

+



repul_field

=



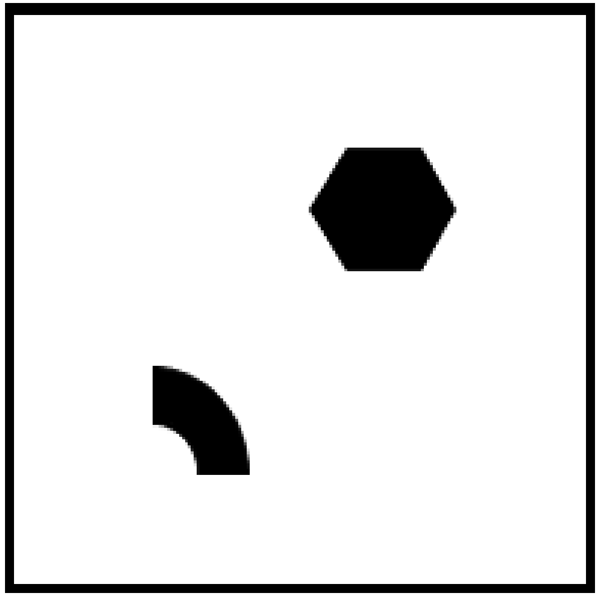
Lots of local minima!

potential

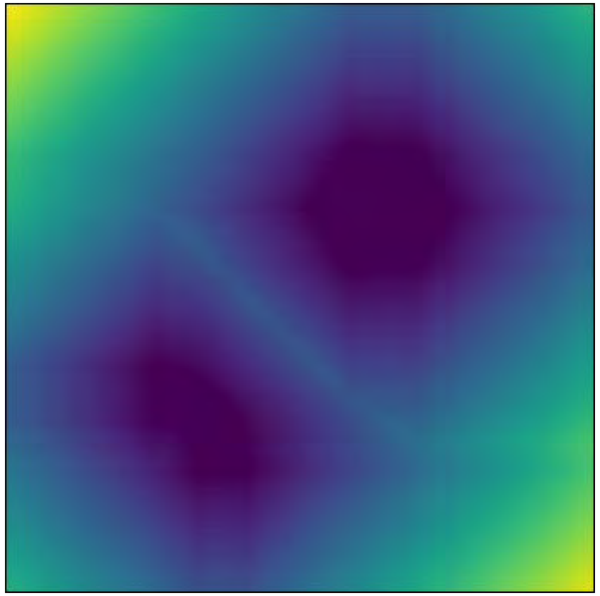
The Repulsion Potential

Idea: Apply a function to the distances to control the shape of the field.

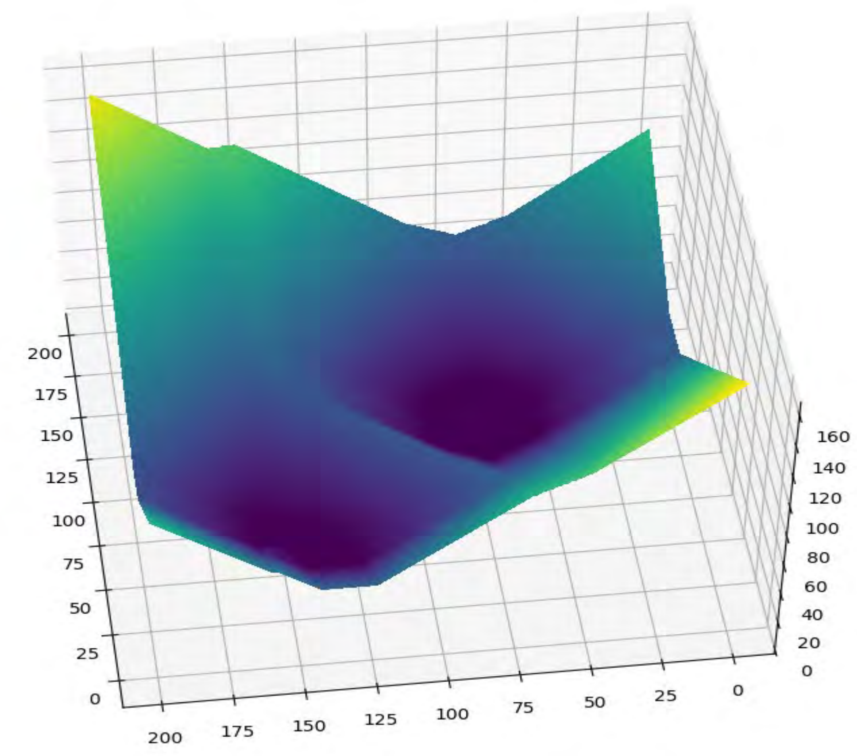
```
repul_field[i] = f(obstacle_dists[i])
```



Binary Image



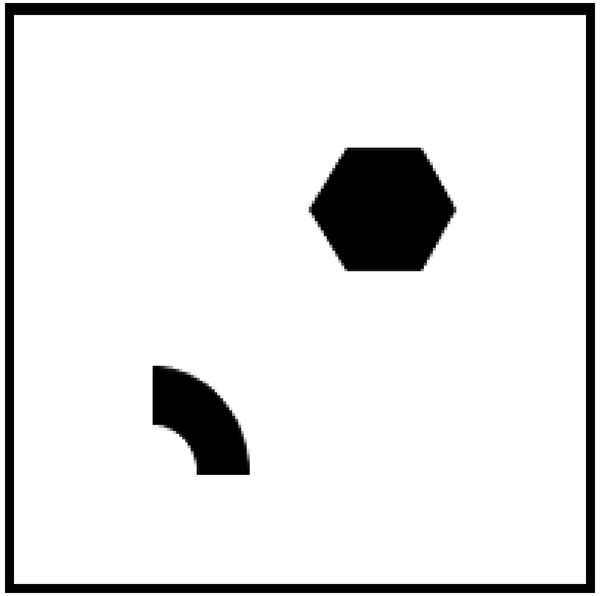
`obstacle_dists`



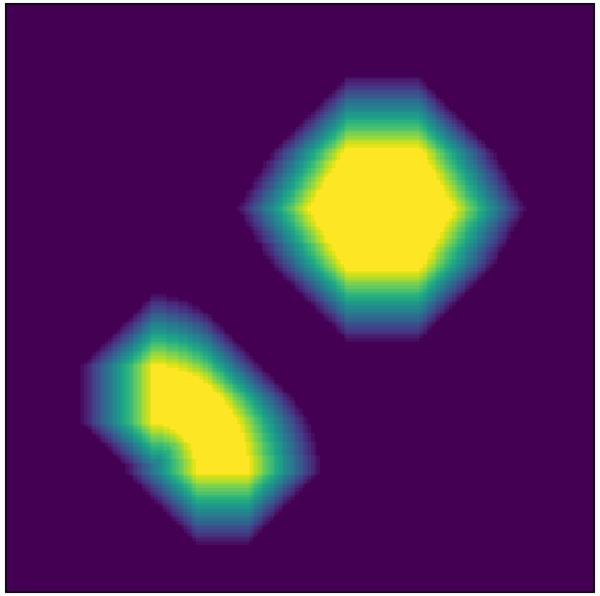
The Repulsion Potential

Idea: Threshold the obstacle distances then take the negative.

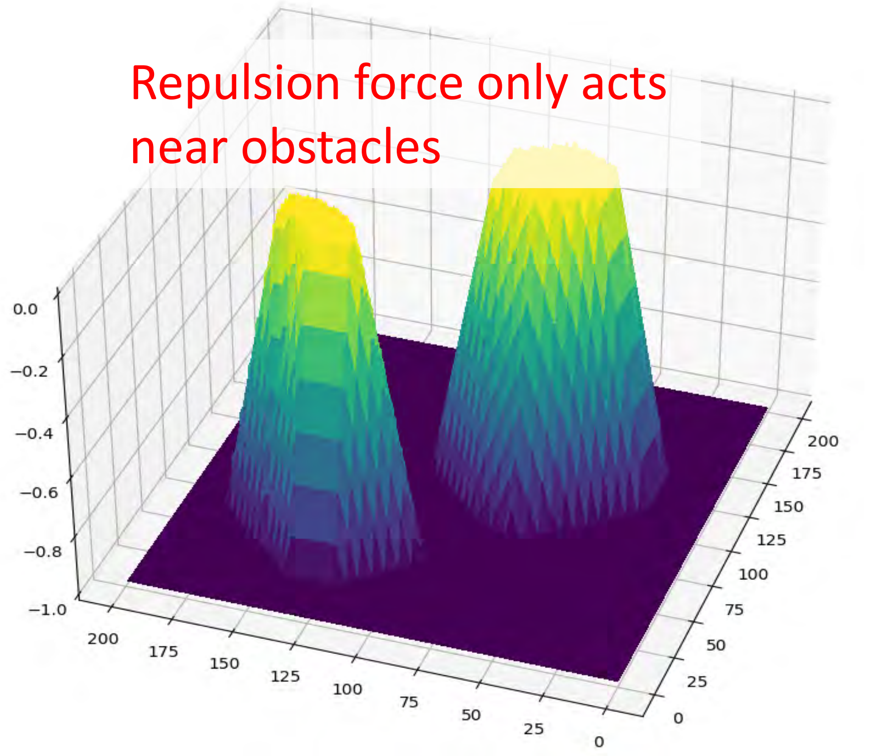
```
repul_field[i] = -min(obstacle_dists[i], THRESH)
```



Binary Image



`repul_field`

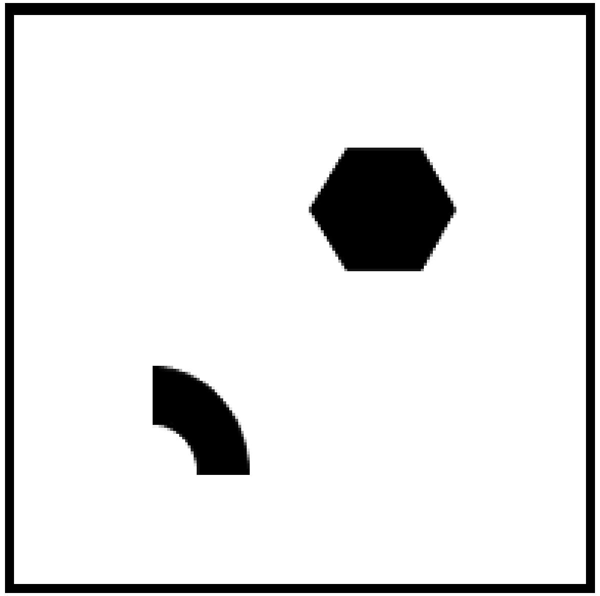


Repulsion force only acts near obstacles

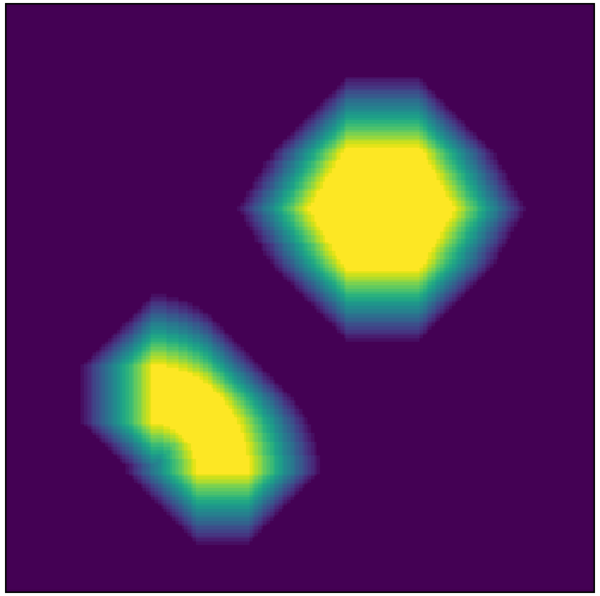
The Repulsion Potential

Idea: Normalize the potential so its range doesn't depend on threshold.

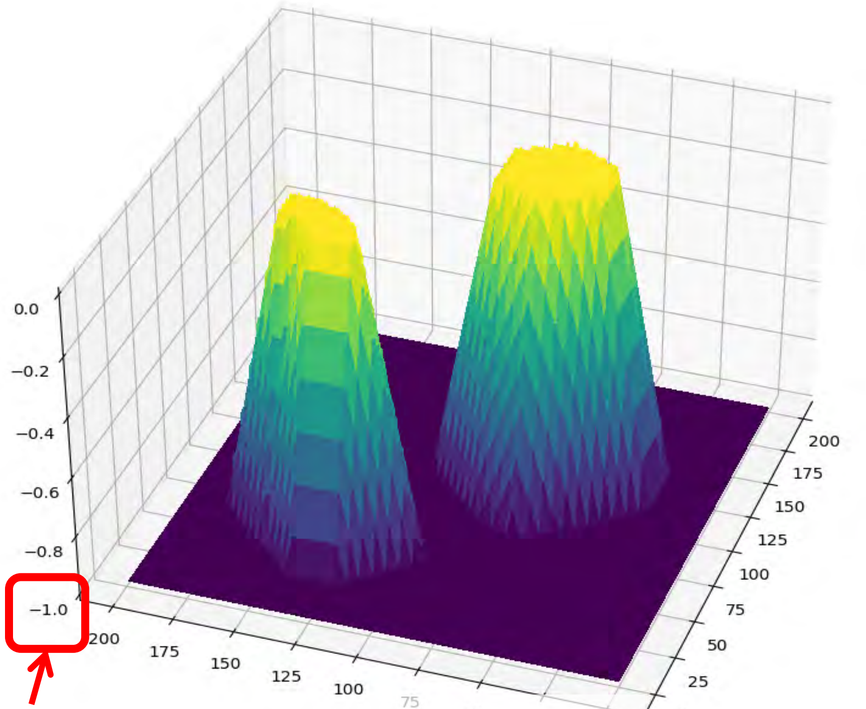
```
repul_field[i] = -min(obstacle_dists[i], THRESH) / THRESH
```



Binary Image



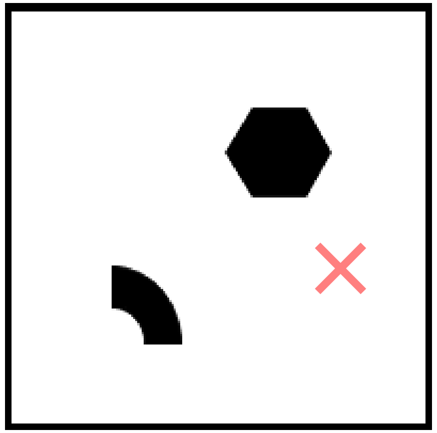
repul_field



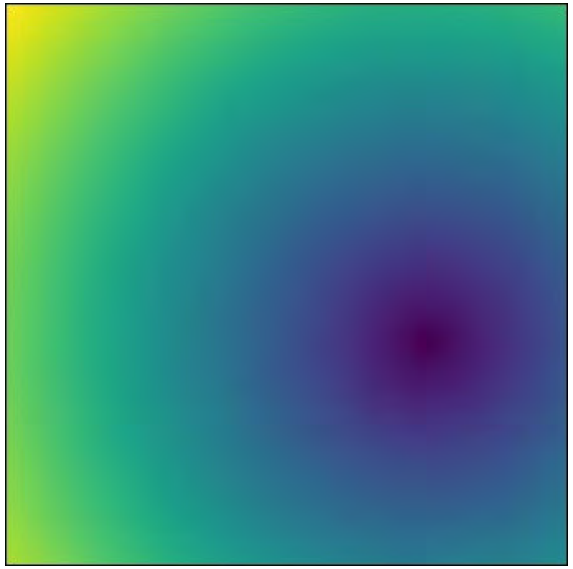
All values are between -1 and 0

Combining Potentials

two_obstacles.map

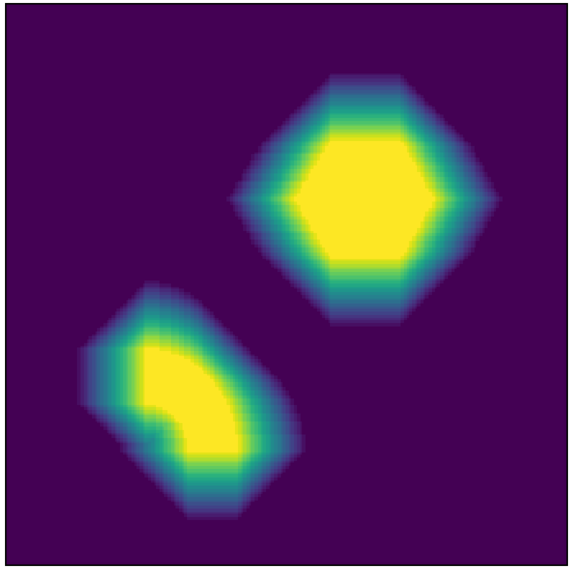


Let’s combine this new potential with our “cone” attractive potential:



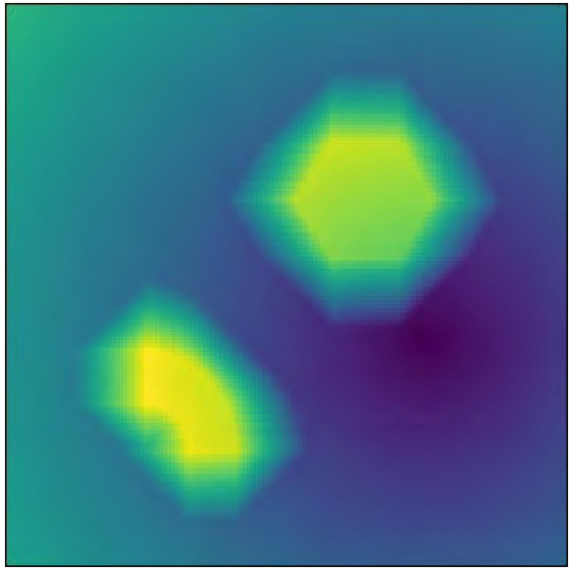
attract_field

+



repul_field

=

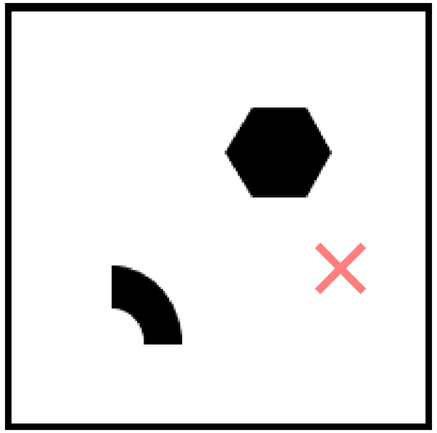


potential

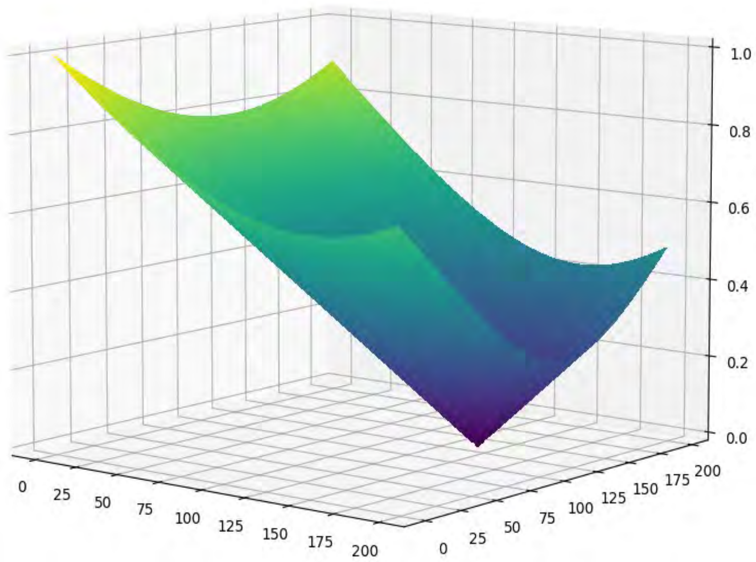
`goal_dists / max(goal_dists)` ← Normalize this too (values from 0 to 1)

Combining Potentials

two_obstacles.map

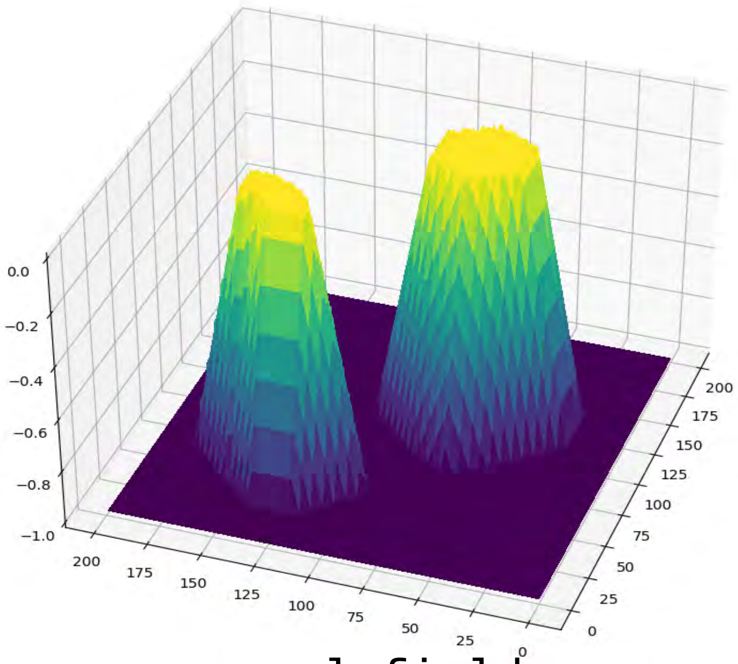


Let's combine this new potential with our "cone" attractive potential:



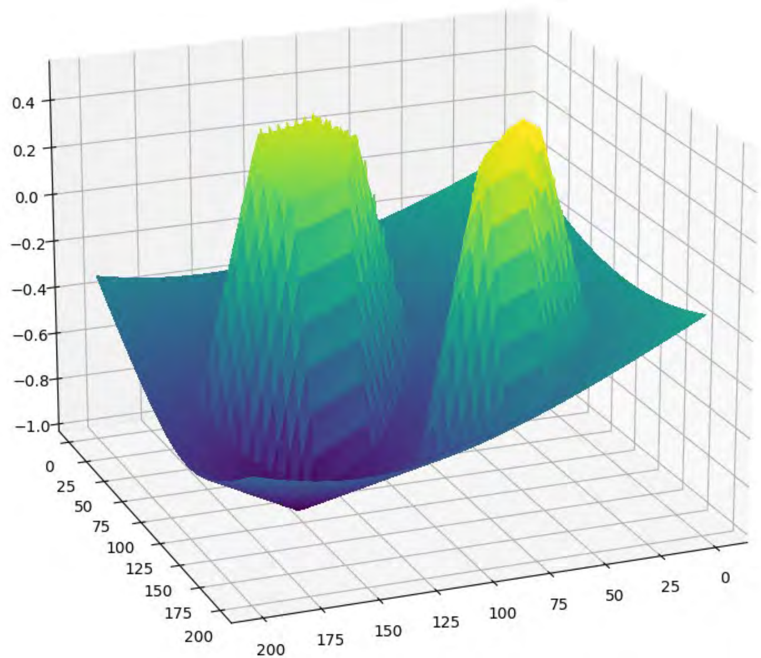
attract_field

+



repul_field

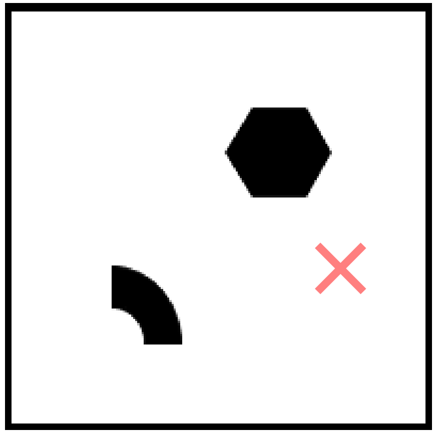
=



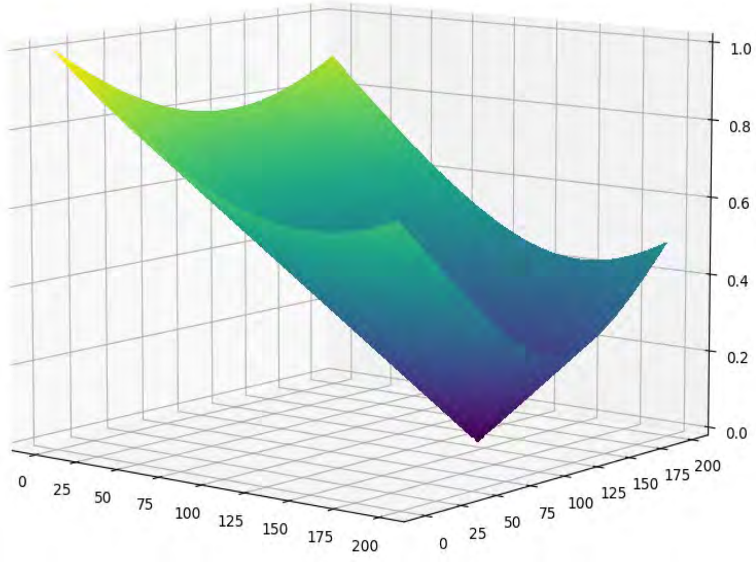
potential

Combining Potentials

two_obstacles.map

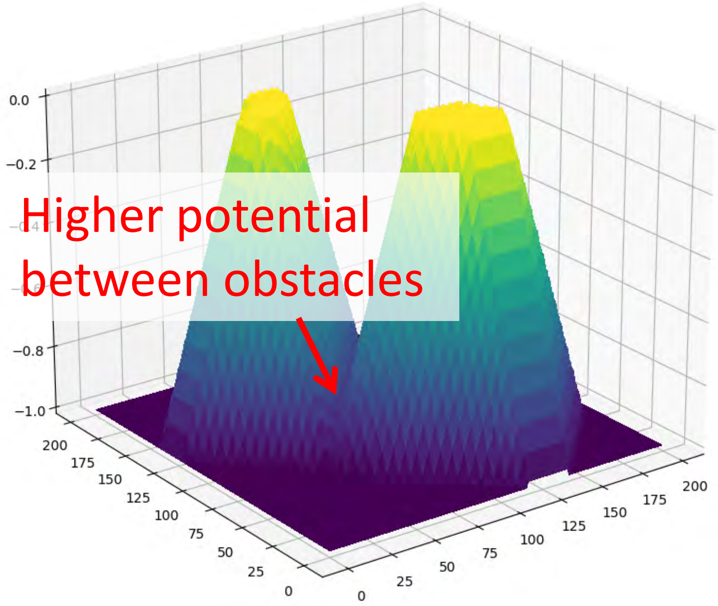


What if we use a higher threshold?



attract_field

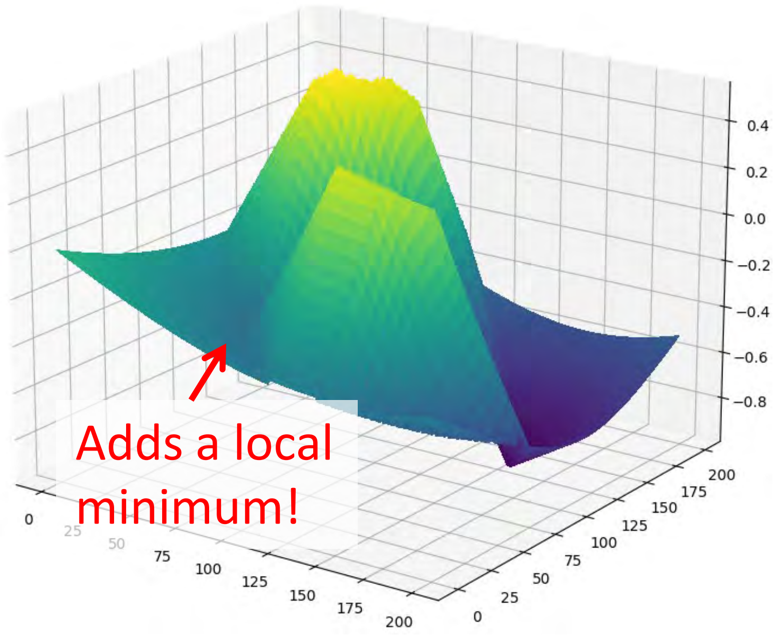
+



Higher potential between obstacles

repul_field

=

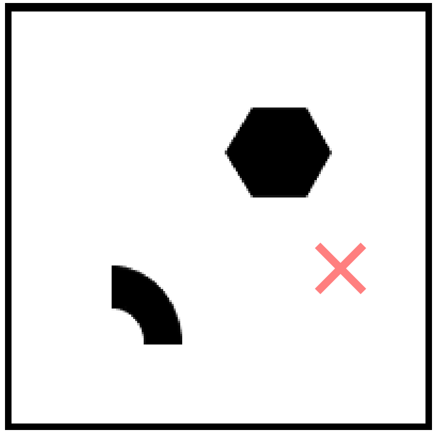


Adds a local minimum!

potential

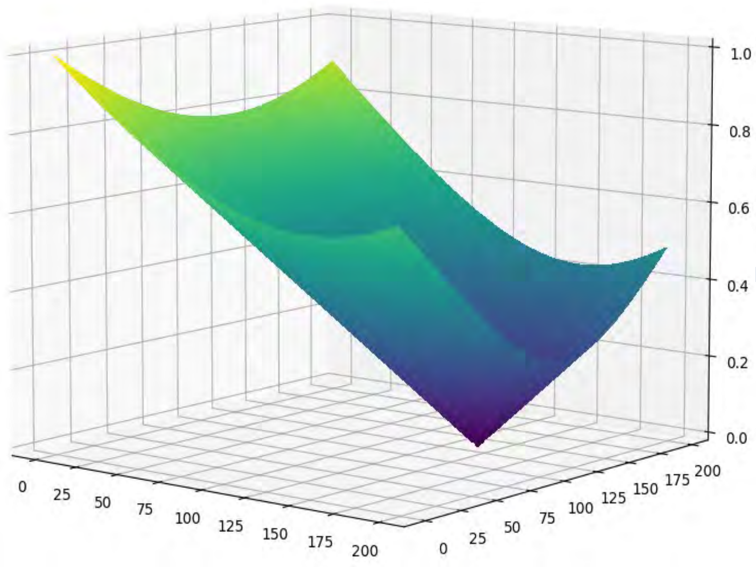
Combining Potentials

two_obstacles.map



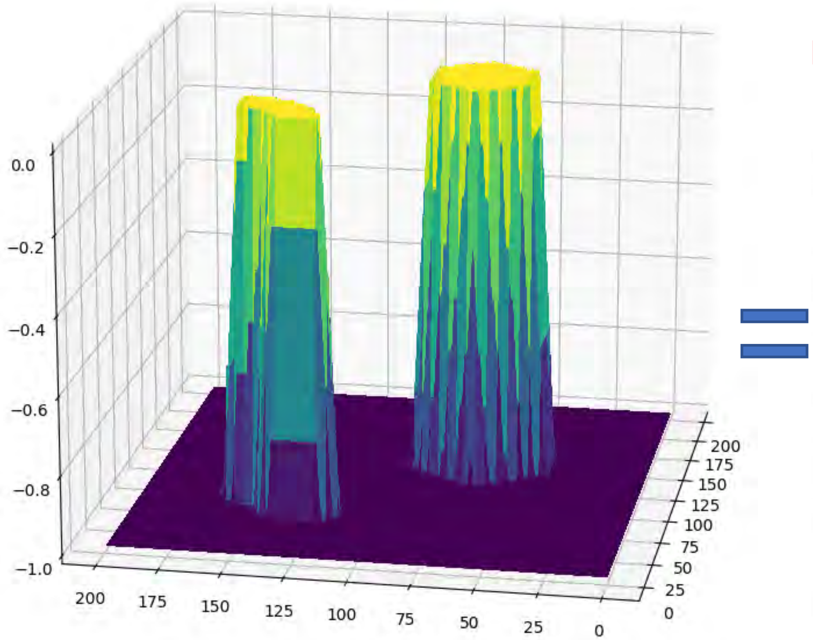
For very small thresholds, the robot might collide with obstacles!

What if we use a lower threshold?



attract_field

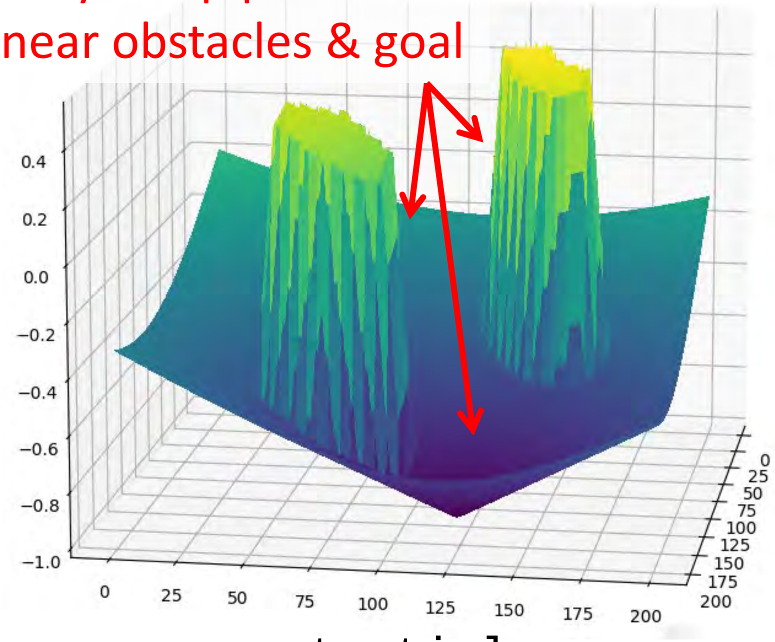
+



repul_field

=

Very steep potential near obstacles & goal

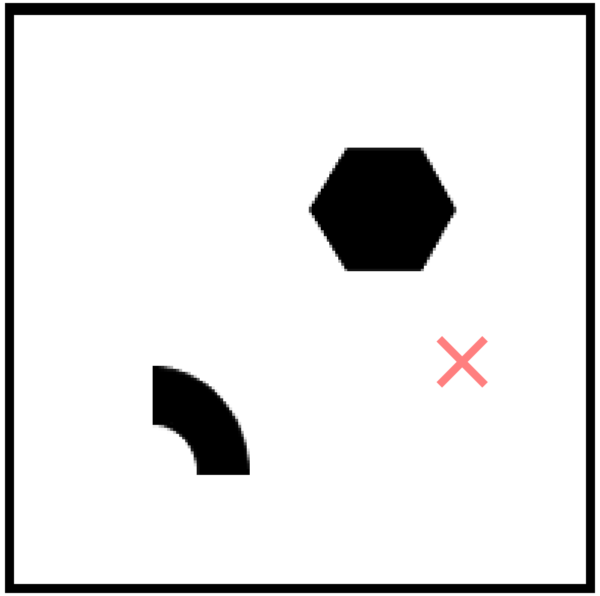


potential

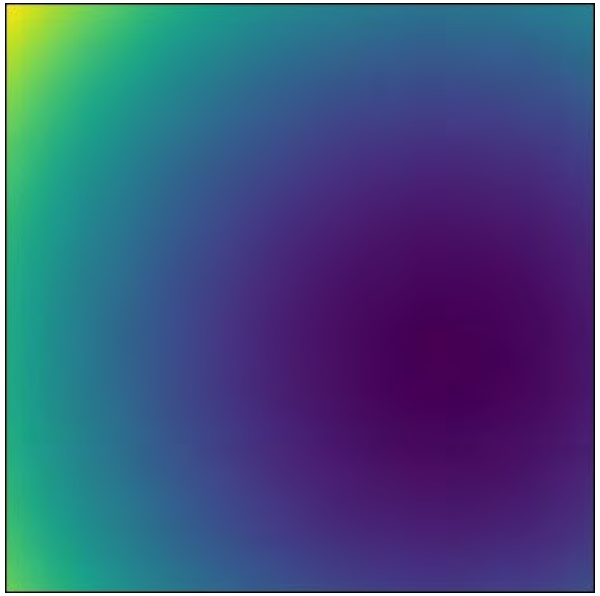
Another Attraction Potential

Idea: Make the slope of the potential steeper far away from the goal and less steep close to the goal.

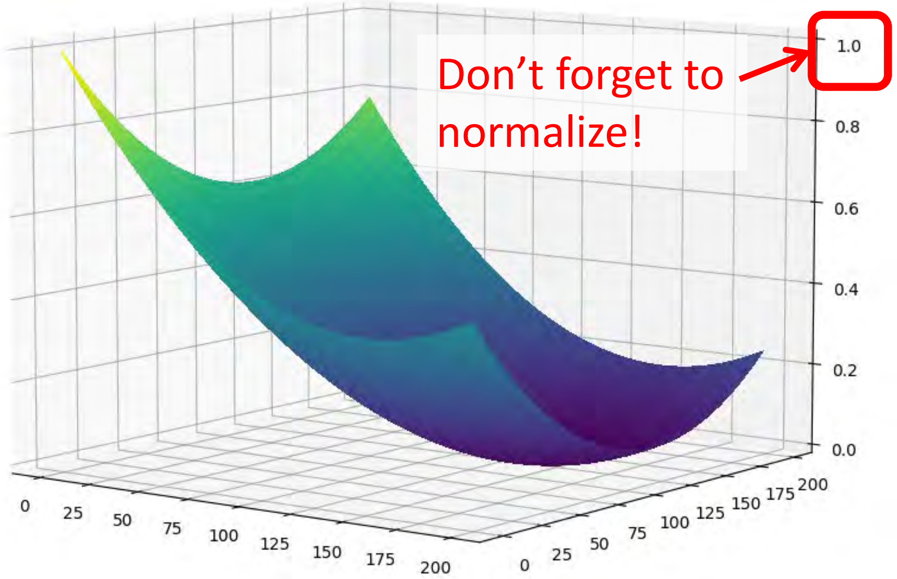
```
attract_field[i] = goal_dists[i] * goal_dists[i]
```



Binary Image



attract_field



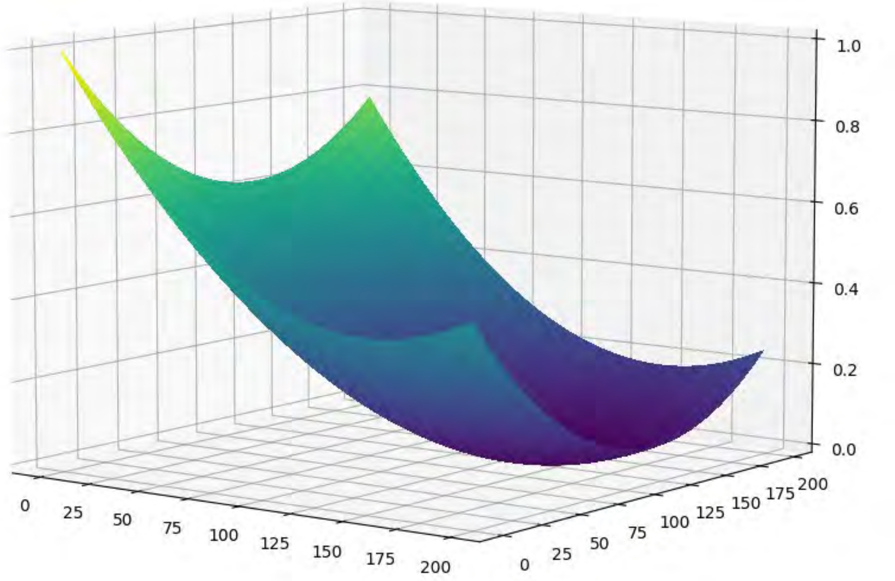
Don’t forget to normalize!

Another Attraction Potential

Idea: Make the slope of the potential steeper far away from the goal and less steep close to the goal.

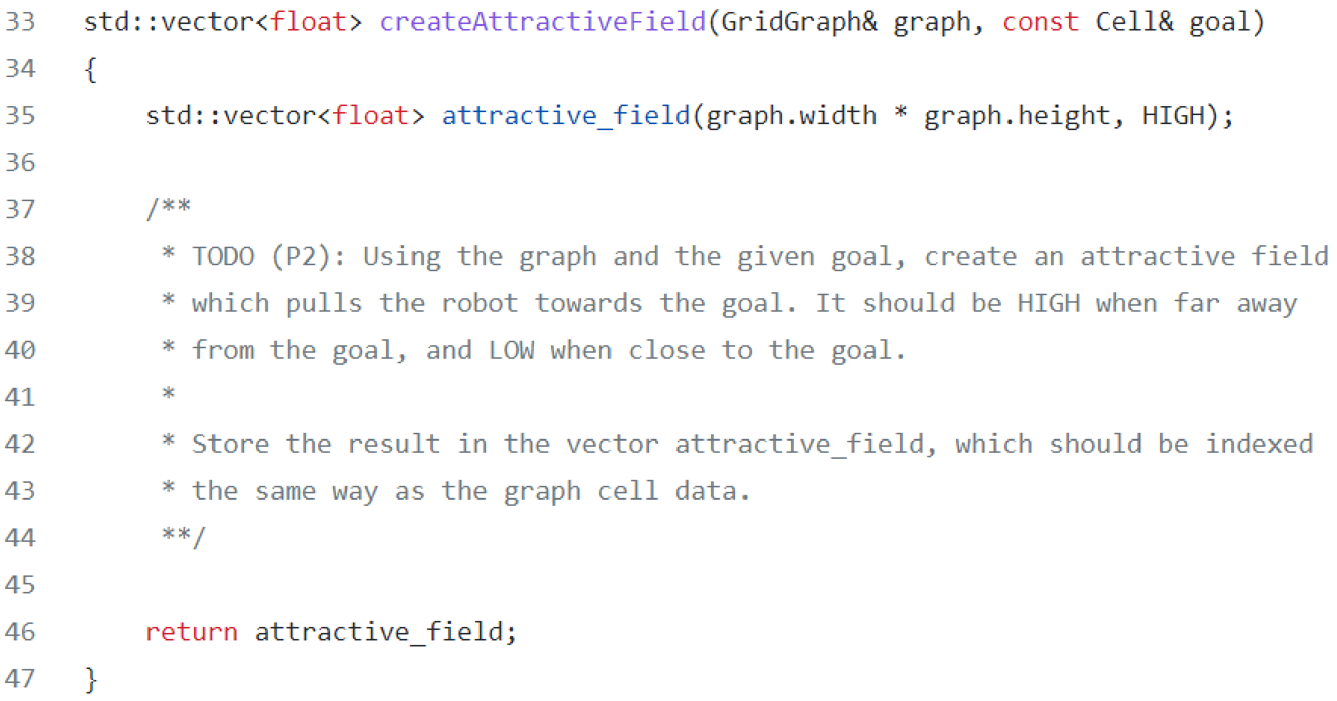
```
attract_field[i] = goal_dists[i] * goal_dists[i]
```

This defines a bowl potential.

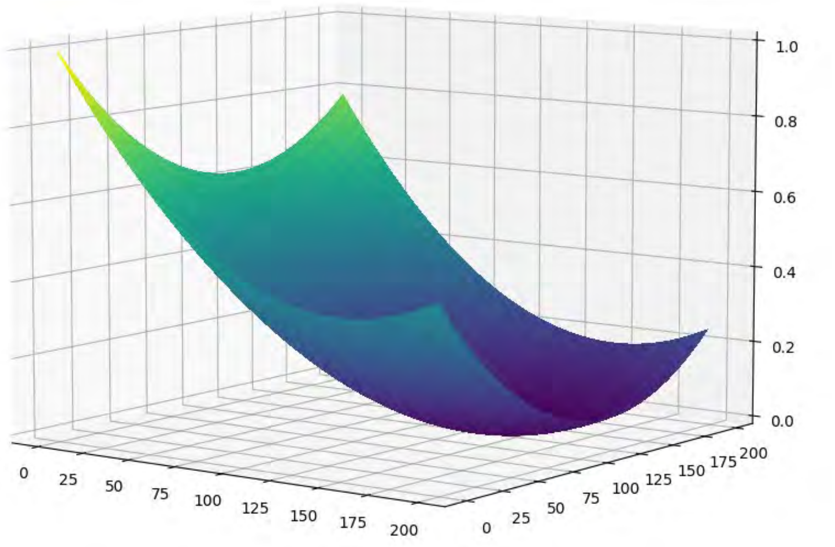


P2.1: Implementing the Attraction Potential

Implement your own version of the attraction potential in the function `createAttractiveField()`. Try different functions!



```cpp
std::vector<float> createAttractiveField(GridGraph& graph, const Cell& goal)
{
    std::vector<float> attractive_field(graph.width * graph.height, HIGH);

    /**
     * TODO (P2): Using the graph and the given goal, create an attractive field
     * which pulls the robot towards the goal. It should be HIGH when far away
     * from the goal, and LOW when close to the goal.
     *
     * Store the result in the vector attractive_field, which should be indexed
     * the same way as the graph cell data.
     **/

    return attractive_field;
}
```

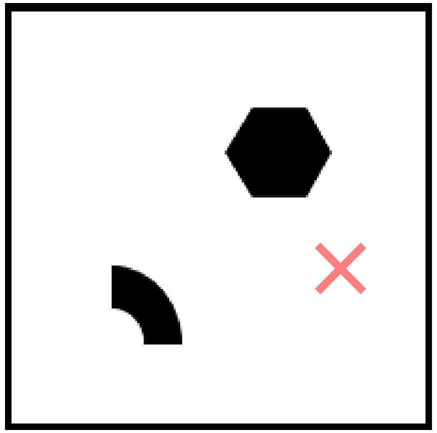


src/potential_field/potential_field.cpp

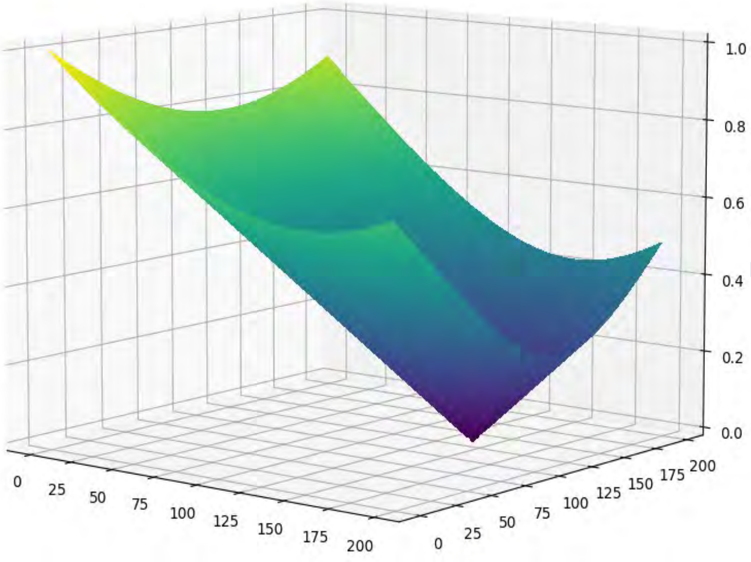
Combining Potentials

For very small thresholds, the robot might collide with obstacles!

two_obstacles.map

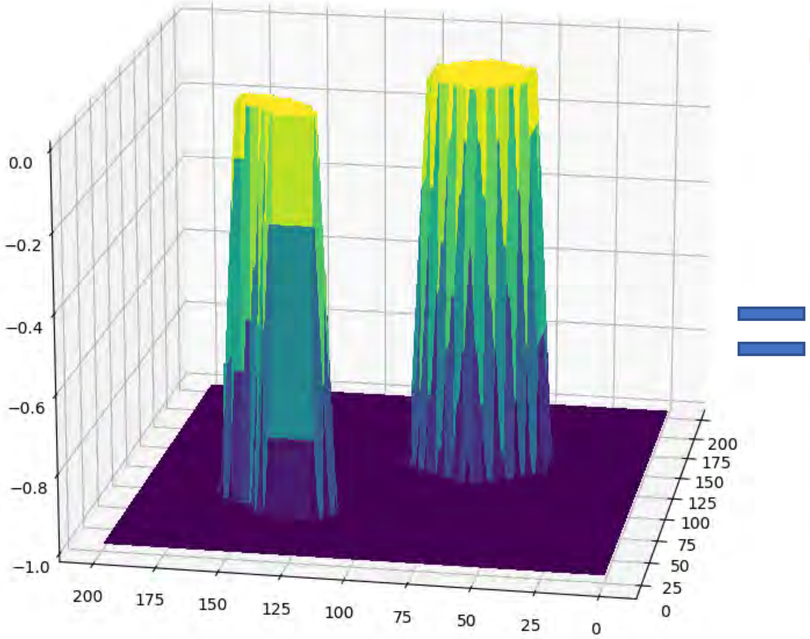


What if we use a lower threshold?



attract_field

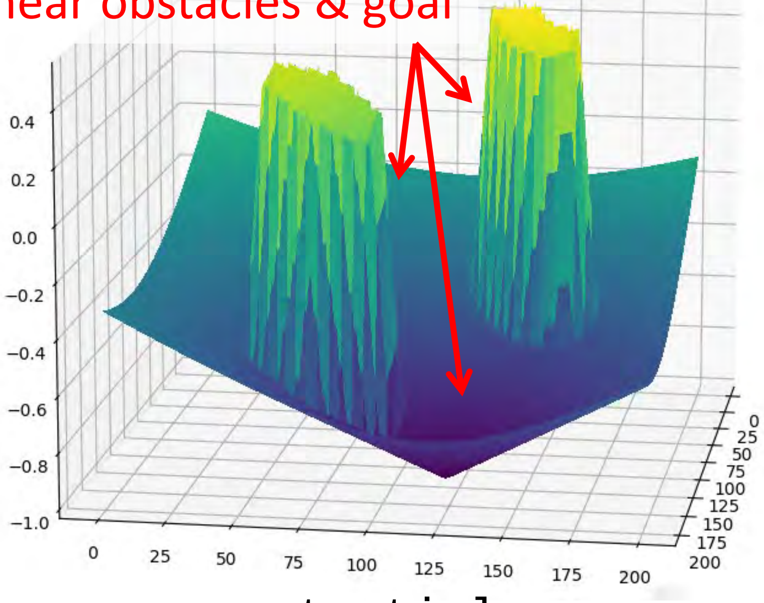
+



repul_field

=

Very steep potential near obstacles & goal



potential

The Exponential Function

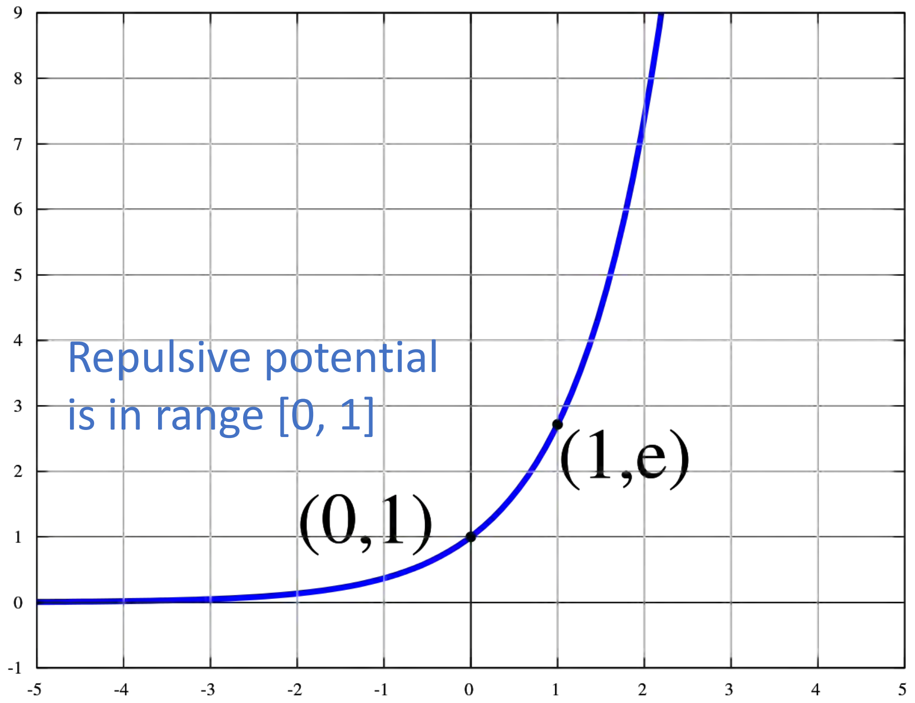
The exponential function can squash (negative) values between 0 and 1. The slope of the field is steeper closer to obstacles and lower far away from obstacles.

```
repul_field[i] = exp(-obstacle_dists[i])
```

The exponential function: e^x

e is Euler’s number.

Negative distances are in range $[-\infty, 0]$

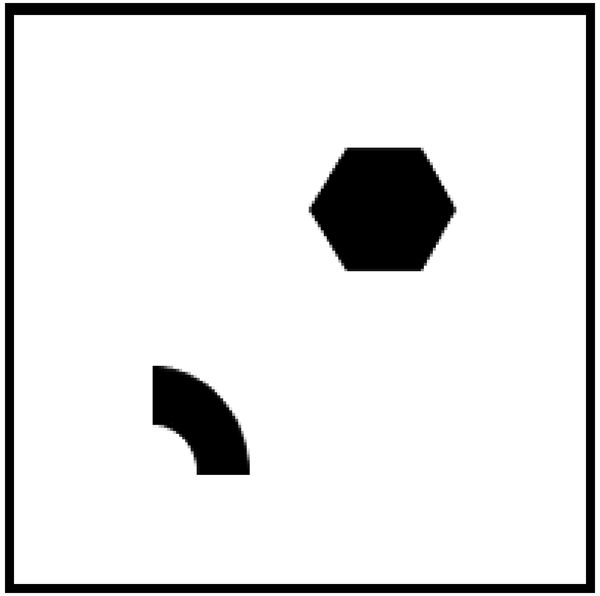


Repulsive potential is in range [0, 1]

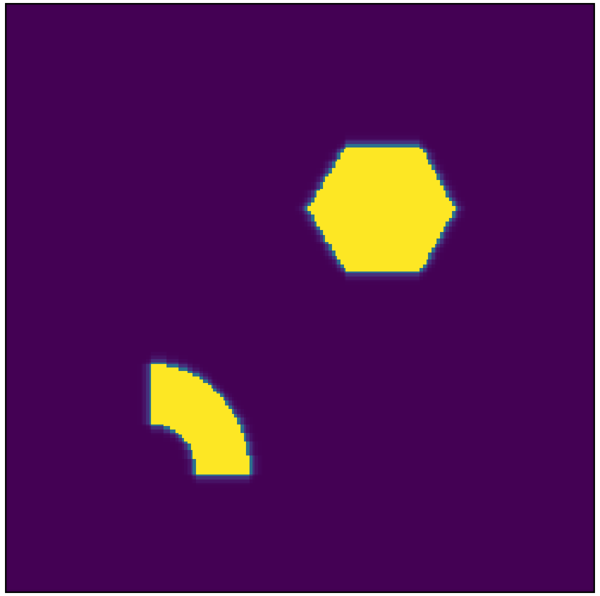
Another Repulsion Potential

Idea: Apply the exponential function to the negative of the distance transform.

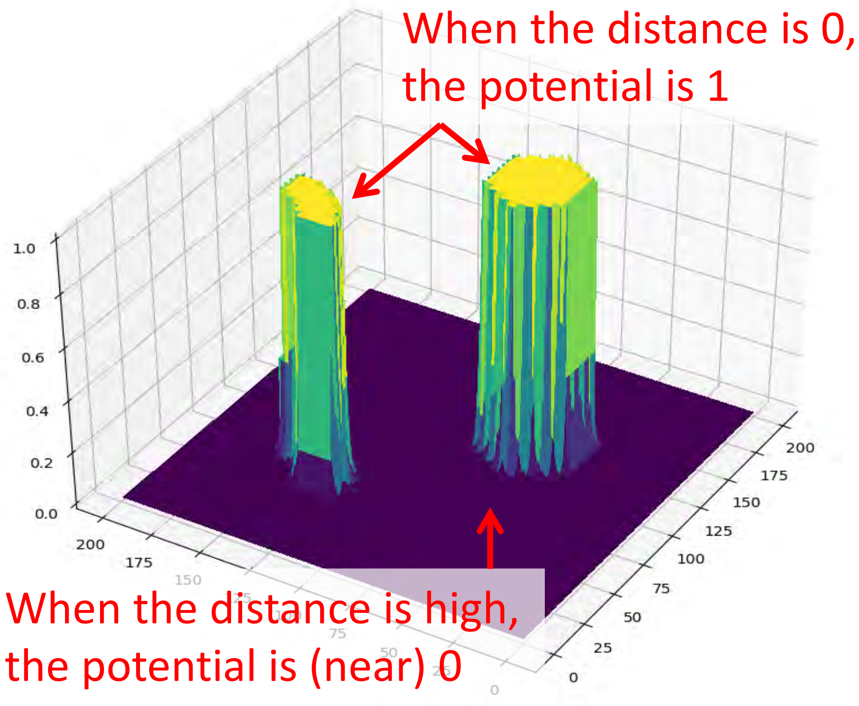
```
repul_field[i] = exp(-obstacle_dists[i])
```



Binary Image



`repul_field`

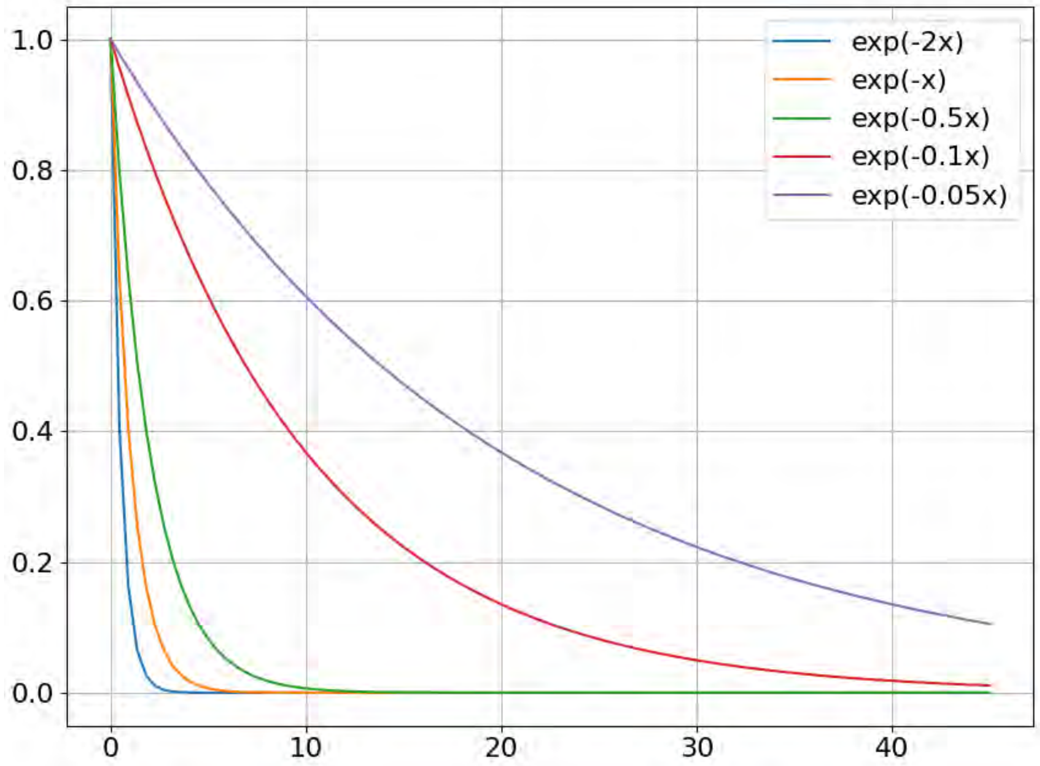


When the distance is 0, the potential is 1

When the distance is high, the potential is (near) 0

The Exponential Function

We can control the steepness of the potential by multiplying the distances by a coefficient.



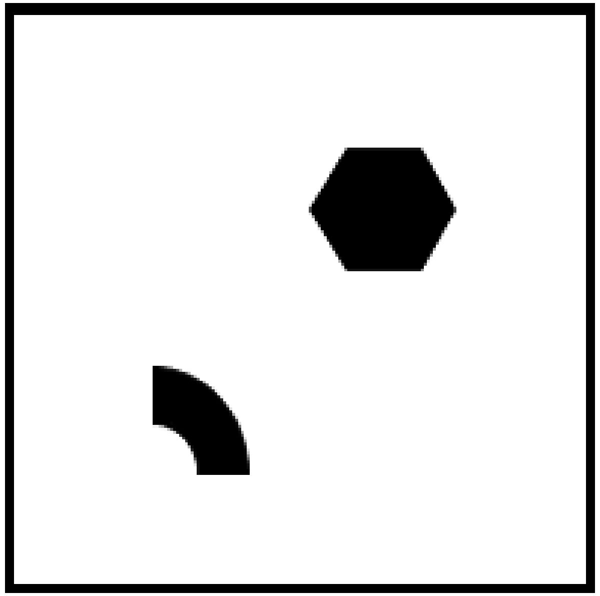
coefficient

```
repul_field[i] = exp(-C * obstacle_dists[i])
```

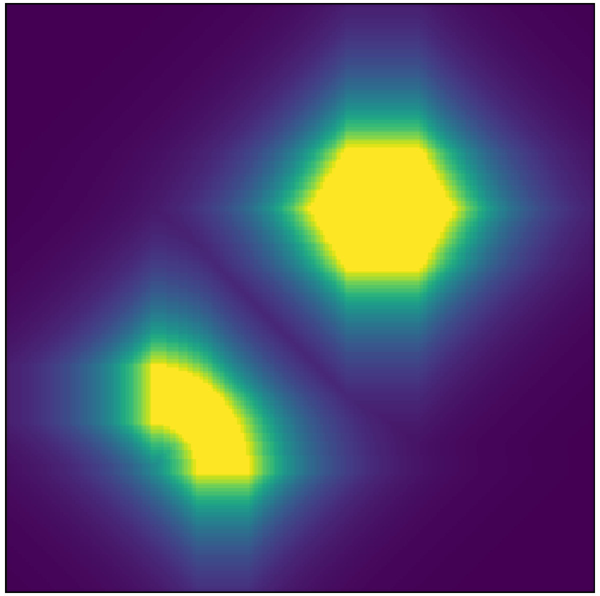
Another Repulsion Potential

Idea: Apply the exponential function to the negative of the distance transform *multiplied by a coefficient*.

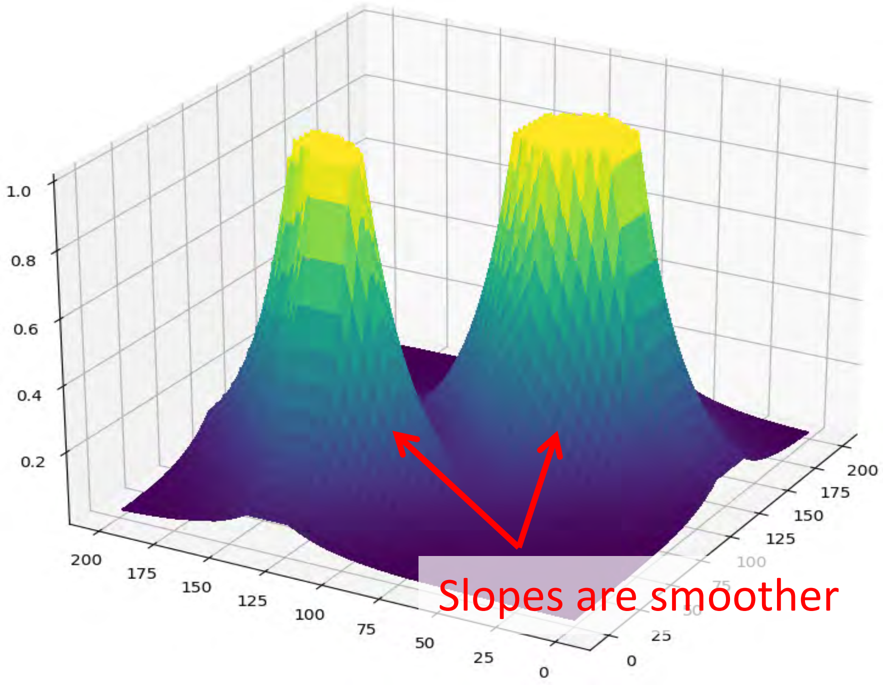
```
repul_field[i] = exp(-C * obstacle_dists[i])
```



Binary Image



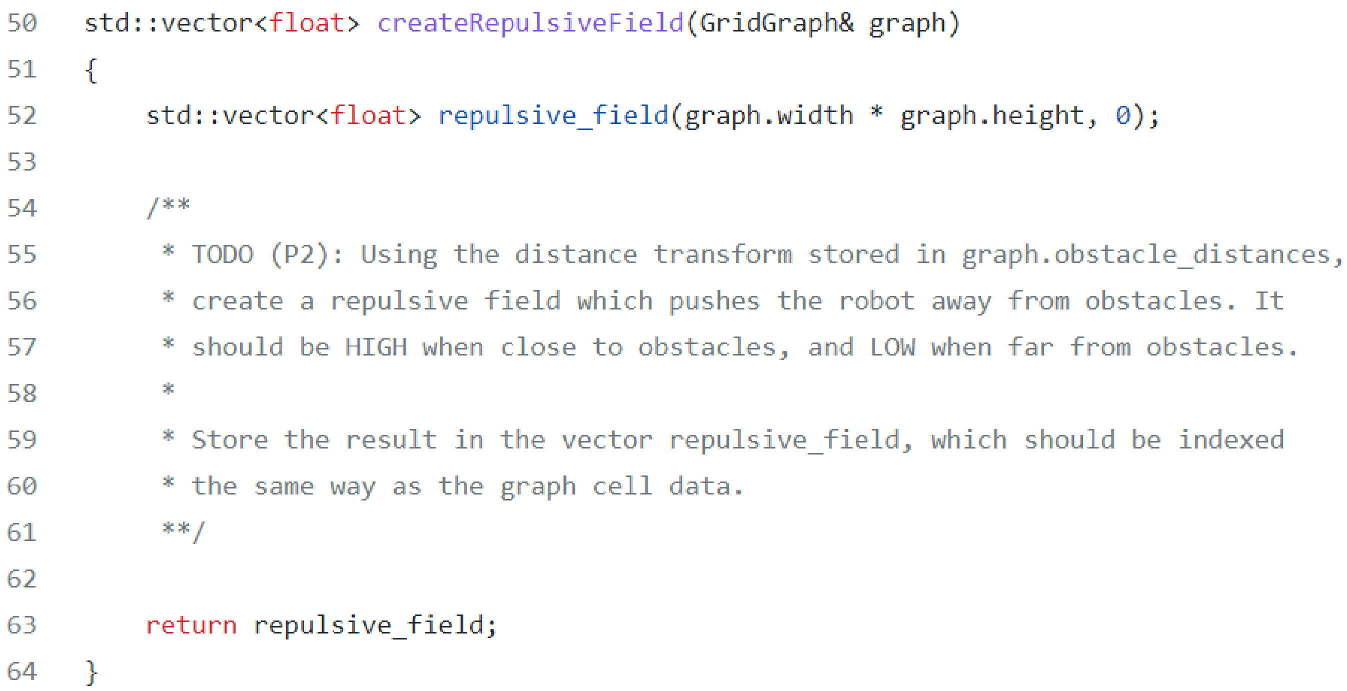
repul_field



Slopes are smoother

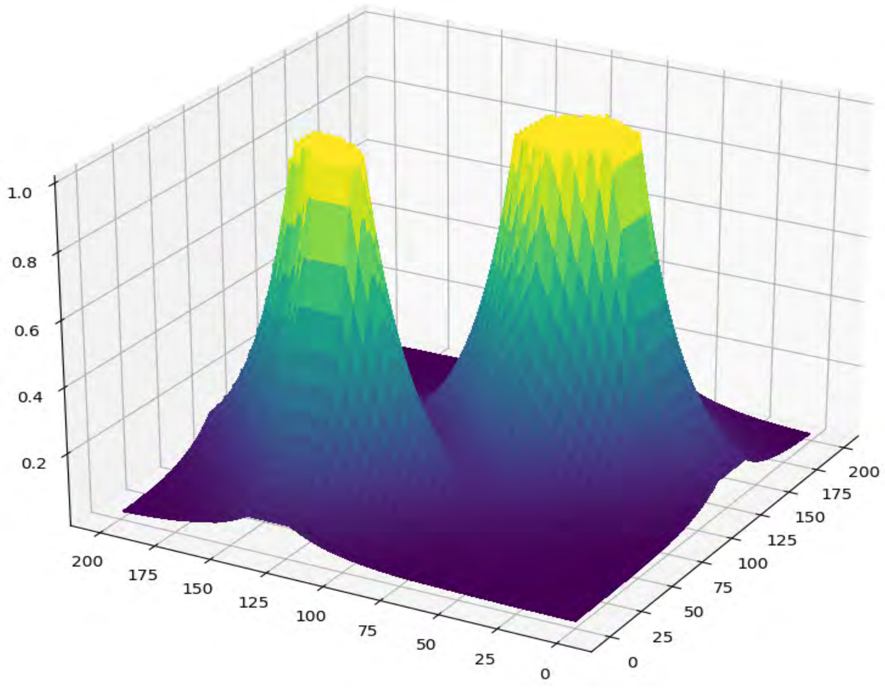
P2.3: Implementing the Repulsion Potential

Implement your own version of the repulsion potential in the function `createRepulsiveField()`. Try different functions!



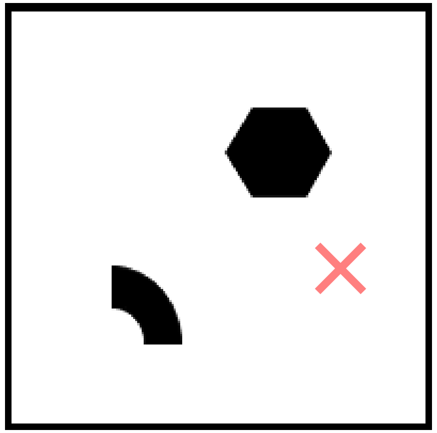
```
std::vector<float> createRepulsiveField(GridGraph& graph)
{
    std::vector<float> repulsive_field(graph.width * graph.height, 0);

    /**
     * TODO (P2): Using the distance transform stored in graph.obstacle_distances,
     * create a repulsive field which pushes the robot away from obstacles. It
     * should be HIGH when close to obstacles, and LOW when far from obstacles.
     *
     * Store the result in the vector repulsive_field, which should be indexed
     * the same way as the graph cell data.
     **/

    return repulsive_field;
}
```

`src/potential_field/potential_field.cpp`

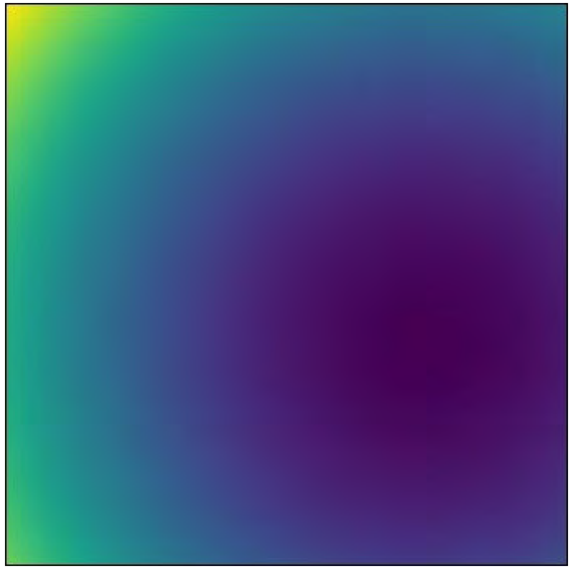


Combining Potentials

two_obstacles.map

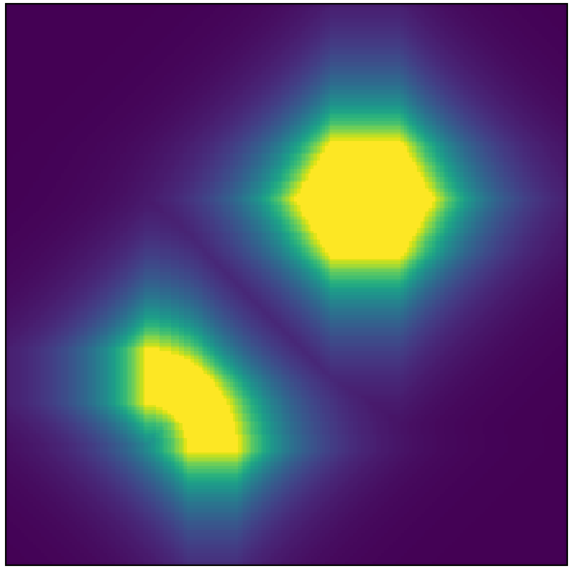


Let’s combine the exponential repulsion potential with our “bowl” attraction potential:



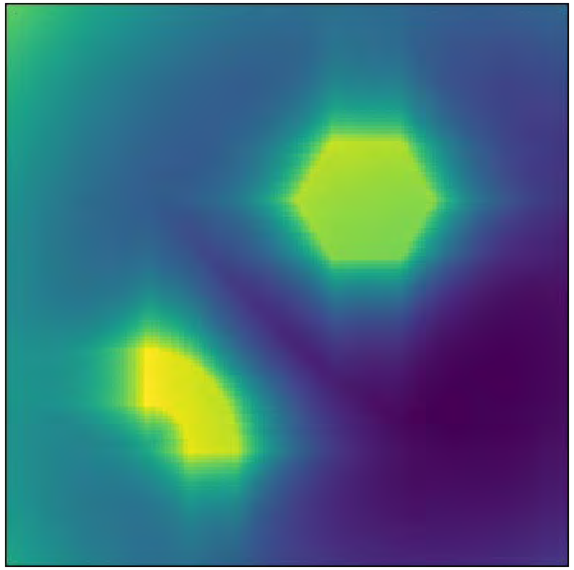
attract_field

+



repul_field

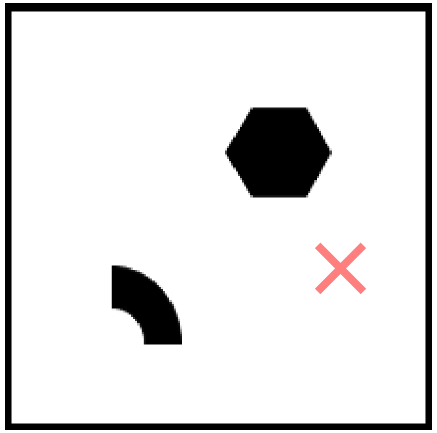
=



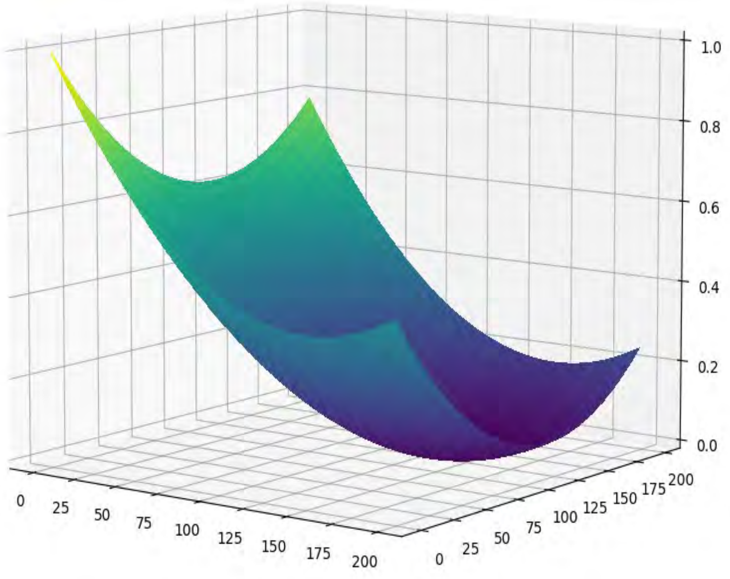
potential

Combining Potentials

two_obstacles.map

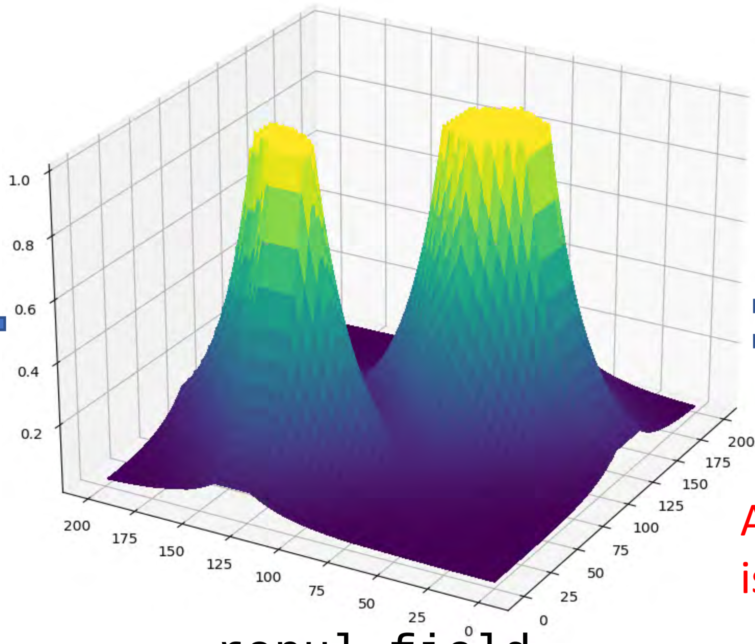


Let’s combine the exponential repulsion potential with our “bowl” attraction potential:



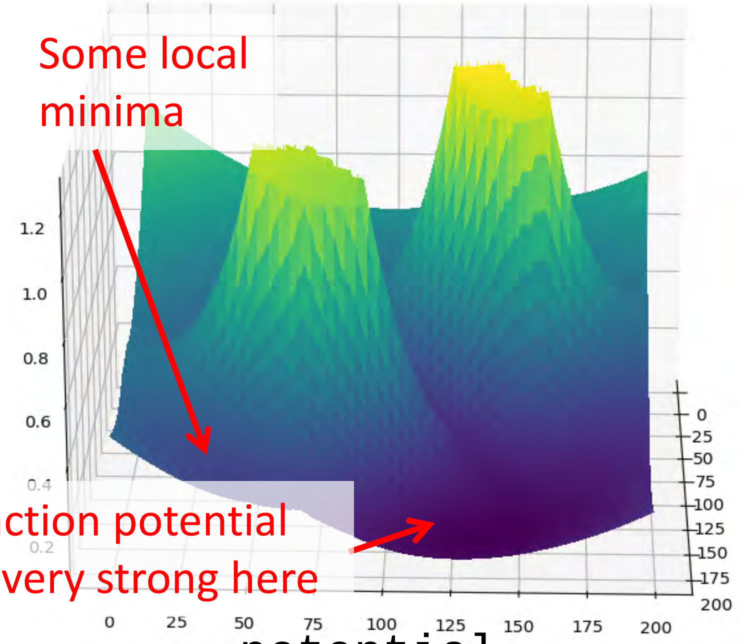
attract_field

+



repul_field

=



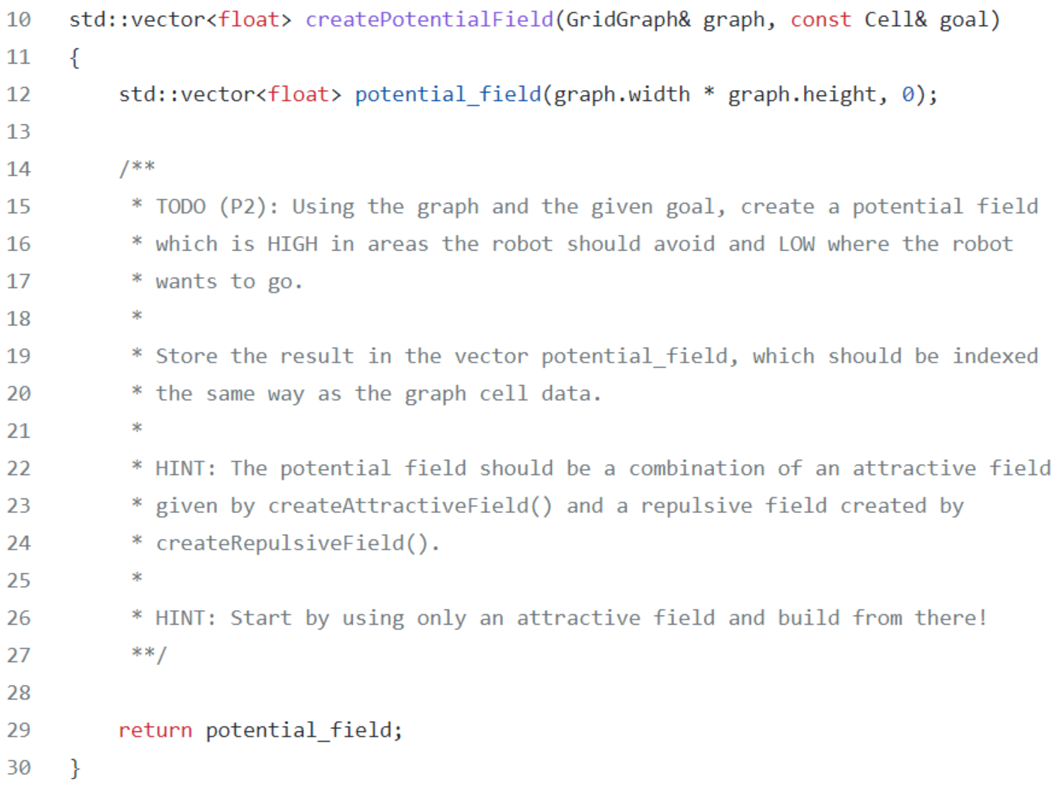
Some local minima

Attraction potential isn’t very strong here

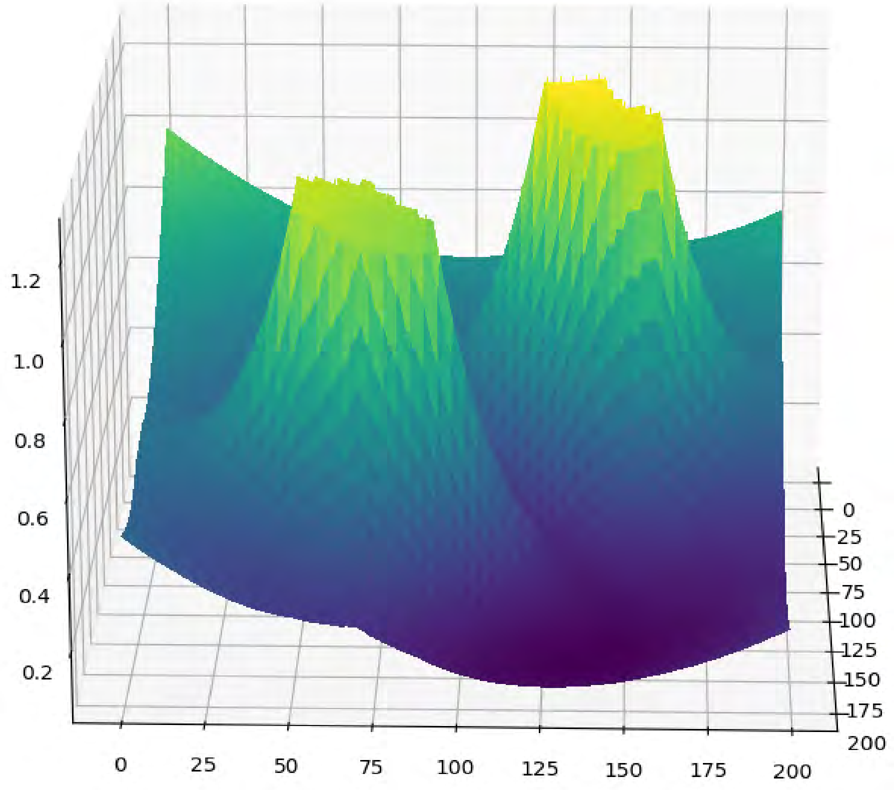
potential

P2.3: Combining Potentials

Combine the attractive and repulsive potentials in function `createPotentialField()`.



```
std::vector<float> createPotentialField(GridGraph& graph, const Cell& goal)
{
    std::vector<float> potential_field(graph.width * graph.height, 0);

    /**
     * TODO (P2): Using the graph and the given goal, create a potential field
     * which is HIGH in areas the robot should avoid and LOW where the robot
     * wants to go.
     *
     * Store the result in the vector potential_field, which should be indexed
     * the same way as the graph cell data.
     *
     * HINT: The potential field should be a combination of an attractive field
     * given by createAttractiveField() and a repulsive field created by
     * createRepulsiveField().
     *
     * HINT: Start by using only an attractive field and build from there!
     **/

    return potential_field;
}
```

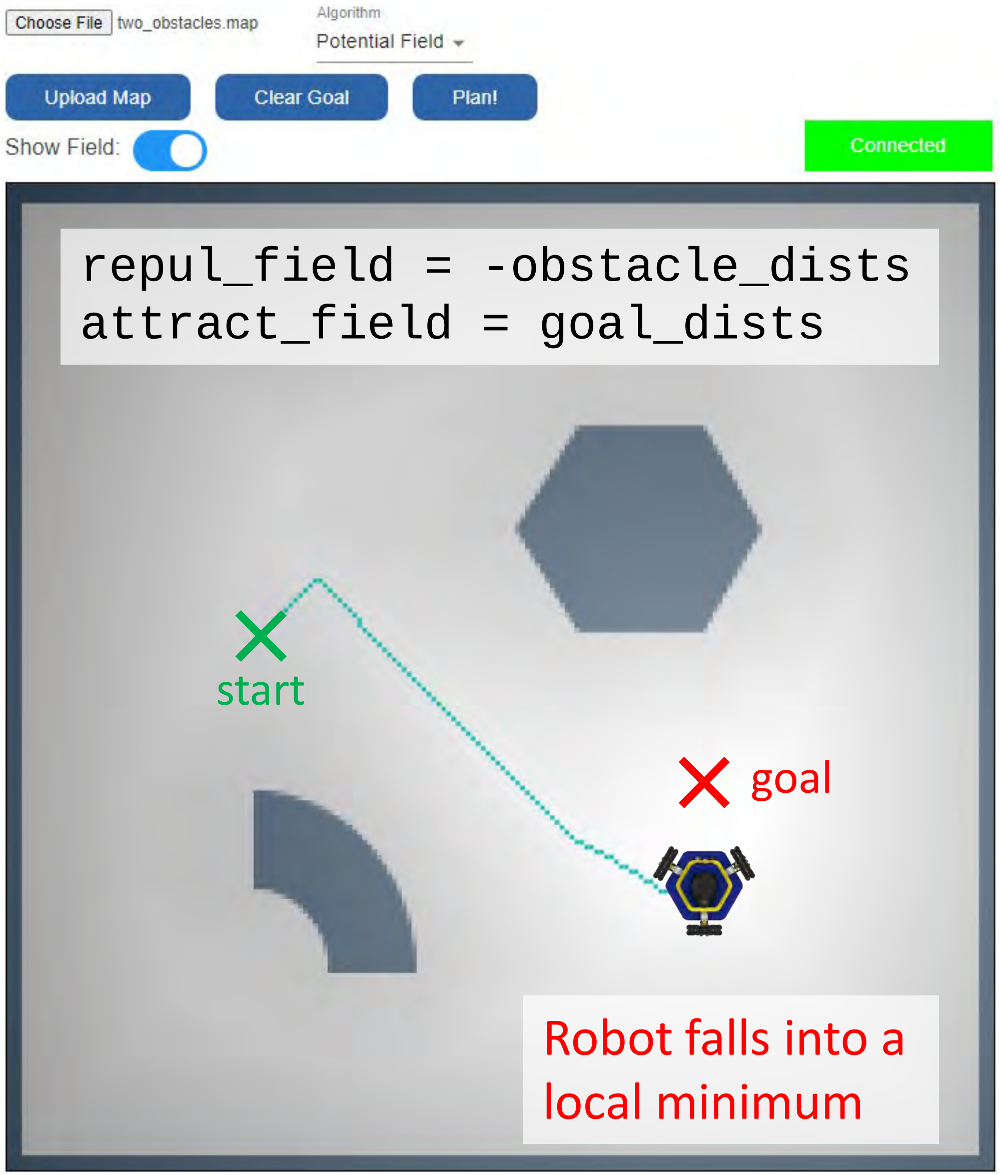


src/potential_field/potential_field.cpp

Potential Field Navigation

Once we have defined a potential function, we can perform local search from any start location.

Local search will navigate to the nearest **local minimum**.



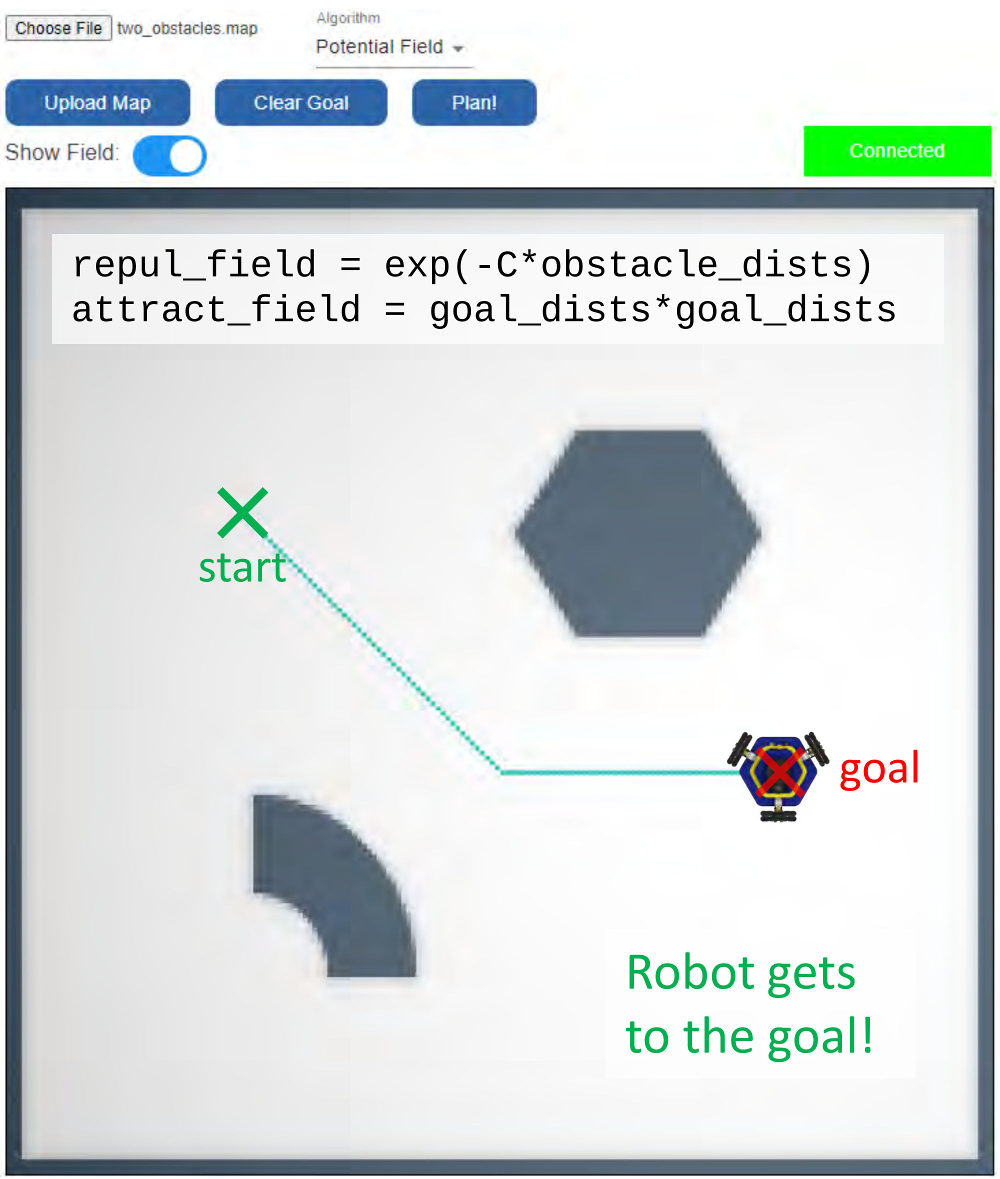
```
repul_field = -obstacle_dists
attract_field = goal_dists
```

Robot falls into a local minimum

Potential Field Navigation

Once we have defined a potential function, we can perform local search from any start location.

Local search will navigate to the nearest **local minimum**.



```
repul_field = exp(-C*obstacle_dists)
attract_field = goal_dists*goal_dists
```

start

goal

Robot gets to the goal!

Tuning the Potential Field

Choosing a good potential function is an art! Try your algorithm in different maps with different start and goal locations.

Your potential function must generalize as much as possible to different maps, start positions, and end goals! On demo day, you will not be able to tune your functions to a specific map.

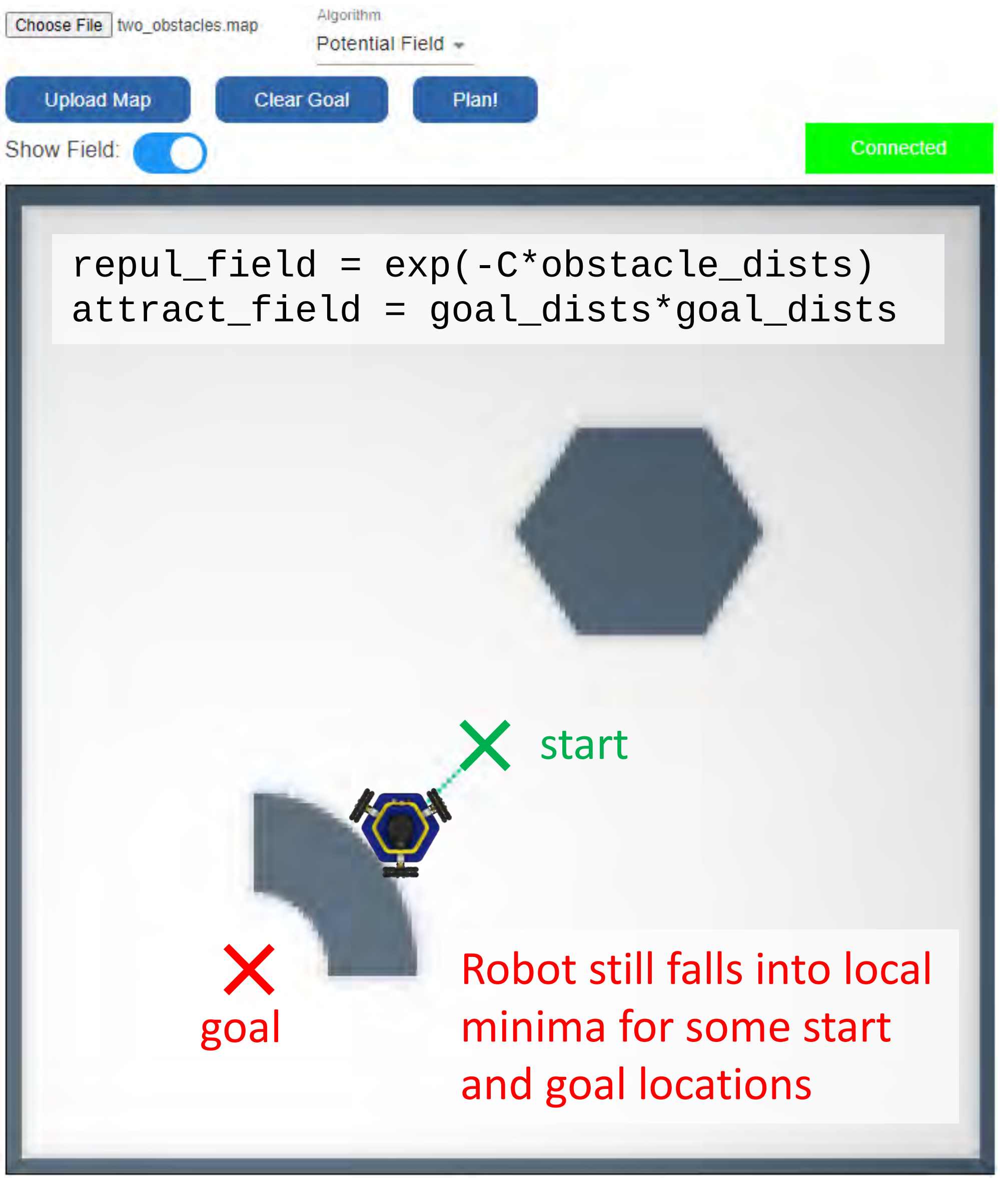
Warning: If your distances are in cells, note that the cells aren't always the same size in all maps! (The cell size is stored in `graph.meters_per_cell`)

Potential Field Navigation

Once we have defined a potential function, we can perform local search from any start location.

Local search will navigate to the nearest **local minimum**.

The navigation web app is helpful for designing your potential functions.



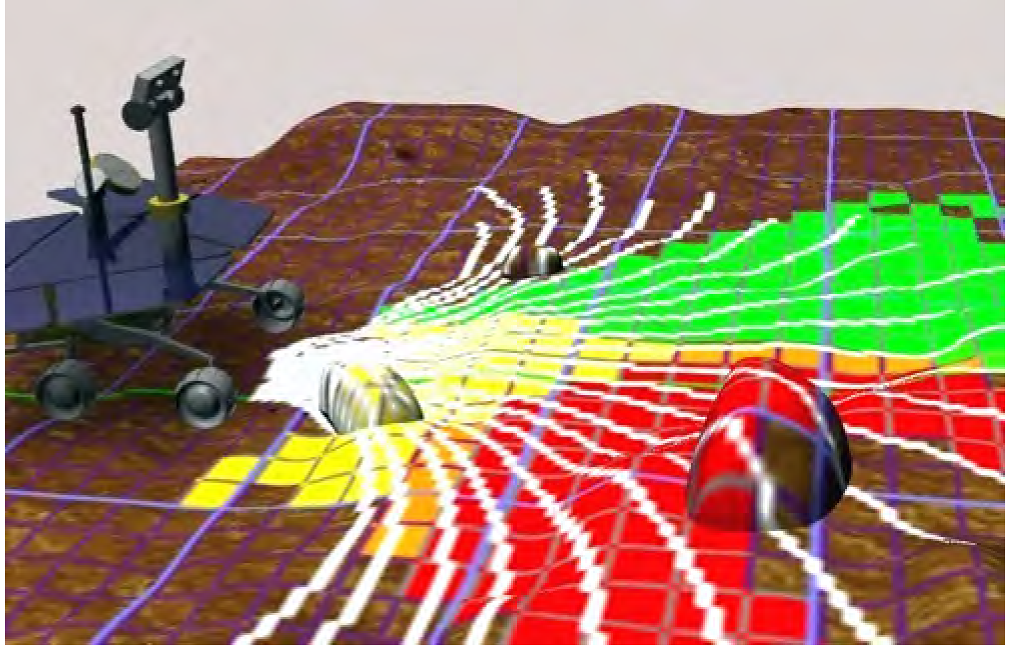
```
repul_field = exp(-C*obstacle_dists)
attract_field = goal_dists*goal_dists
```

Robot still falls into local minima for some start and goal locations

Summary: Potential Field Navigation

Potential field control is used for local navigation, to nearby goals.

Can we use potential field navigation for global navigation?

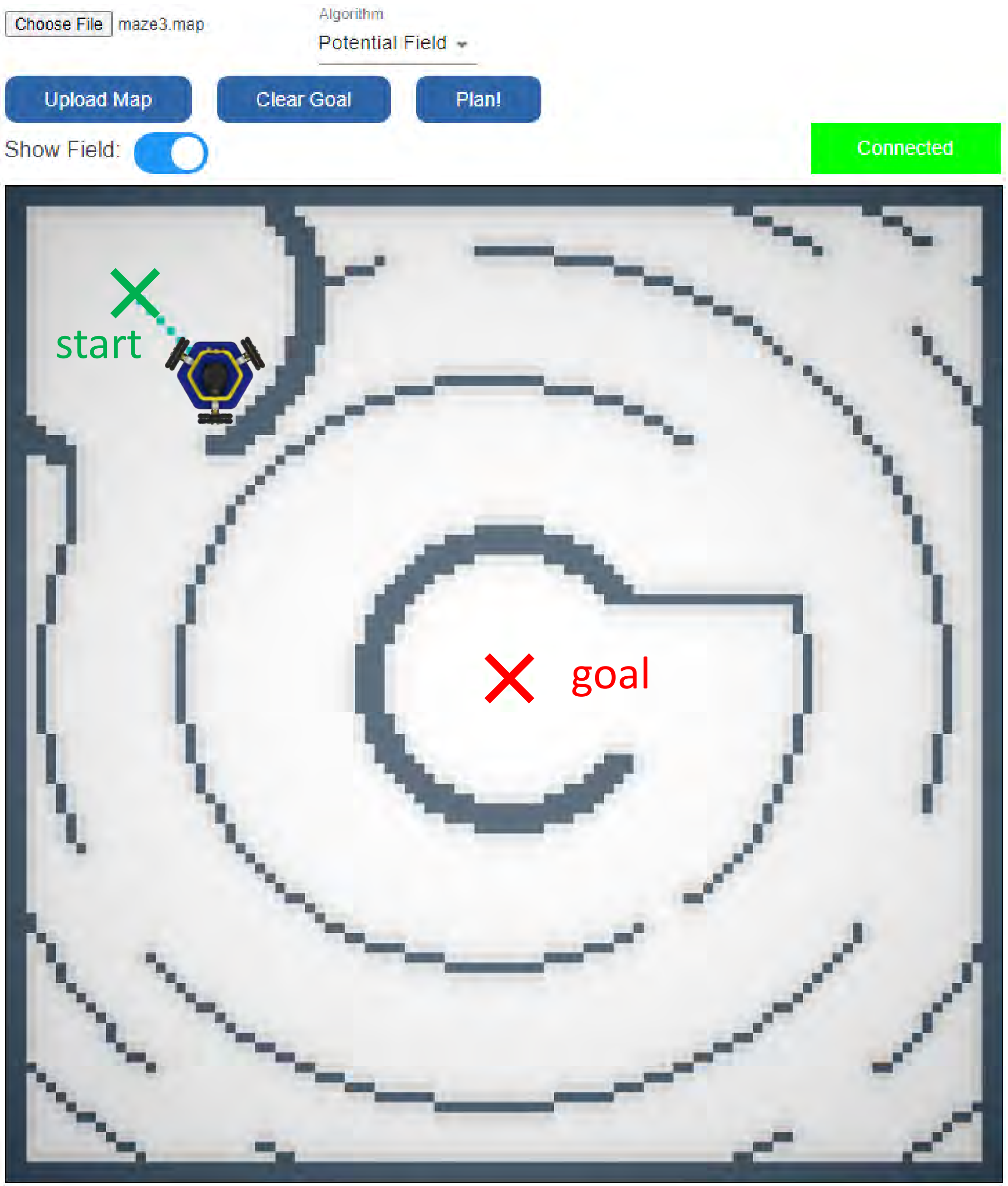


Completeness

Is there a potential field that will allow the robot to navigate to the goal in this map?

A navigation algorithm is **complete** if it always finds a path when one exists, or detects when no path exists.

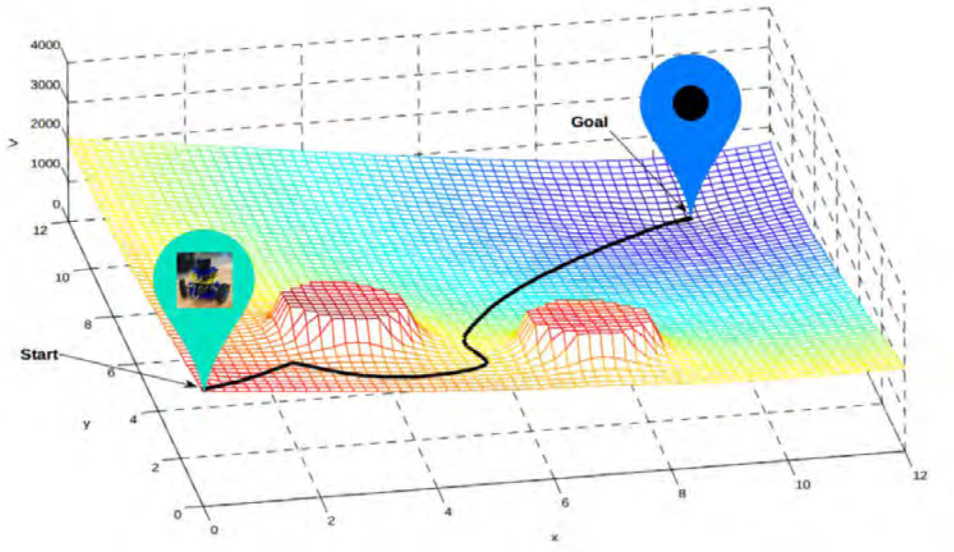
Project 3: Graph search algorithms for robot navigation.



Project 2: Potential Field Navigation

- ☑ Build a map of environment
- ☑ Form attraction potential to goal
- ☑ Form repulsion potentials away from obstacles
- ☑ Add potentials together into potential field
- ☑ Local search over potential field to navigate

Today!


y
Start
Goal
x